Detailed Report on the Adventure Works ETL Assignment

-- Sayan Das

1. Data Extraction

I began by loading the data from the provided Excel files. The data consisted of six dimension tables (DimCustomer, DimEmployee, DimGeography, DimProduct, DimReseller, and DimSalesTerritory) and two fact tables (FactInternetSales and FactResellerSales). In this phase, I ensured that the correct data types were applied to each column by specifying the `dtype` argument and using `parse_dates` for date columns.

Code Snippet: Loading Dimension Tables

```
print("Loading dimension tables...")

dim_customer_df = pd.read_excel(
    'DimTables.xlsx',
    sheet_name='DimCustomer',
    dtype={
        'CustomerKey': 'Int64',
        'GeographyKey': 'Int64',
        'CustomerName': 'string',
        'BirthDate': 'string',
        'MaritalStatus': 'string',
        'Gender': 'string',
        'EmailAddress': 'string',
        'YearlyIncome': 'float',
        'Education': 'string',
        'Occupation': 'string',
        'HouseOwnerFlag': 'string',
        'Address': 'string',
        'FirstPurchaseDate': 'string'
    }
)
```

#Similar for all other tables

Code Snippet: Loading Fact Tables

```
print("Loading fact tables...")

fact_internet_sales_df = pd.read_excel(
    'FactInternetSales.xlsx',
    dtype={
        'ProductKey': 'Int64',
        'CustomerKey': 'Int64',
        'SalesTerritoryKey': 'Int64',
        'SalesOrderNumber': 'string',
        'SalesOrderLineNumber': 'Int64',
        'DiscountAmount': 'float',
        'TotalProductCost': 'float',
        'SalesAmount': 'float',
        'Freight': 'float',
        'CarrierTrackingNumber': 'string',  # Will remove
```

```
    },
    parse_dates=['OrderDate', 'DueDate', 'ShipDate']
)

fact_reseller_sales_df = pd.read_excel(
    'FactResellerSales.xlsx',
    dtype={
        'ProductKey': 'Int64',
        'ResellerKey': 'Int64',
        'EmployeeKey': 'Int64',
        'SalesTerritoryKey': 'Int64',
        'SalesOrderNumber': 'string',
        'SalesOrderLineNumber': 'Int64',
        'DiscountAmount': 'float',
        'TotalProductCost': 'float',
        'SalesAmount': 'float',
        'Freight': 'float',
        'CarrierTrackingNumber': 'string',
    },
    parse_dates=['OrderDate', 'DueDate', 'ShipDate']
)
```

2. Data Transformation

After extraction, I performed several key transformations:

- **Removal of Unnecessary Columns:**
  I removed the `CarrierTrackingNumber` column from the FactInternetSales table since it was not fully empty.

  ```
  if 'CarrierTrackingNumber' in fact_internet_sales_df.columns:
      fact_internet_sales_df.drop(columns=['CarrierTrackingNumber'], inplace=True)
      print("Removed 'CarrierTrackingNumber' column from FactInternetSales.")
  ```

- **Handling Missing Data:**
  I applied specific rules to handle missing values:

  - For **DimProduct**, missing values in the `Color` column were replaced with `'NA'`.
  - For **DimSalesTerritory**, any row with missing data was dropped.
  - For the fact tables (**FactInternetSales** and **FactResellerSales**), rows with any missing values were dropped.

  ```
  # DimProduct: fill missing 'Color' with 'NA'
  if 'Color' in dim_product_df.columns:
      missing_color = dim_product_df['Color'].isnull().sum()
      if missing_color > 0:
          print(f"Filling {missing_color} missing 'Color' cells in DimProduct with 
  'NA'.")
          dim_product_df['Color'] = dim_product_df['Color'].fillna('NA')

  # DimSalesTerritory: drop rows with any missing values
  before_dst = dim_salesterritory_df.shape[0]
  dim_salesterritory_df.dropna(inplace=True)
  after_dst = dim_salesterritory_df.shape[0]
  print(f"Dropped {before_dst - after_dst} row(s) from DimSalesTerritory due to 
  missing data.")
  ```

```python
# Fact tables: drop rows with any missing values
before_fis = fact_internet_sales_df.shape[0]
fact_internet_sales_df.dropna(inplace=True)
print(f"Dropped {before_fis - fact_internet_sales_df.shape[0]} row(s) from
FactInternetSales due to missing data.")

before_frs = fact_reseller_sales_df.shape[0]
fact_reseller_sales_df.dropna(inplace=True)
print(f"Dropped {before_frs - fact_reseller_sales_df.shape[0]} row(s) from
FactResellerSales due to missing data.")
```

3. Data Quality Checks

To ensure high-quality data, I implemented two key checks:

- **Primary Key Uniqueness:**
  I verified that each dimension table had unique primary keys, dropping duplicate rows when necessary.

```python
def check_uniqueness(df, key_column, table_name):
    before = df.shape[0]
    df_clean = df.drop_duplicates(subset=[key_column])
    after = df_clean.shape[0]
    dropped = before - after
    if dropped > 0:
        print(f"{dropped} duplicate row(s) dropped from {table_name} based on
primary key '{key_column}'.")
    else:
        print(f"All rows in {table_name} have a unique '{key_column}'.")
    return df_clean

print("\n-- Checking PK uniqueness in dimension tables --")
dim_customer_df = check_uniqueness(dim_customer_df, 'CustomerKey',   'DimCustomer')
dim_employee_df = check_uniqueness(dim_employee_df, 'EmployeeKey',   'DimEmployee')
dim_geography_df = check_uniqueness(dim_geography_df, 'GeographyKey',
'DimGeography')
dim_product_df = check_uniqueness(dim_product_df, 'ProductKey', 'DimProduct')
dim_reseller_df = check_uniqueness(dim_reseller_df, 'ResellerKey',   'DimReseller')
dim_salesterritory_df = check_uniqueness(dim_salesterritory_df,
'SalesTerritoryKey', 'DimSalesTerritory')
```

- **Foreign Key Validation:**
  I validated that each foreign key in the fact tables matched a primary key in the corresponding dimension table. Any fact row with an invalid reference was dropped.

```python
def validate_fk(fact_df, dim_df, fact_fk, dim_pk, fact_table_name, dim_table_name):
    valid_ids = set(dim_df[dim_pk].unique())
    before = fact_df.shape[0]
    fact_df_clean = fact_df[fact_df[fact_fk].isin(valid_ids)]
    after = fact_df_clean.shape[0]
    dropped = before - after
    if dropped > 0:
        print(f"{dropped} row(s) dropped from {fact_table_name} due to invalid
'{fact_fk}' not found in {dim_table_name}.")
```

```python
    else:
        print(f"All rows in {fact_table_name} have a valid foreign key
'{fact_fk}'.")
    return fact_df_clean

print("\n-- Validating foreign keys in FactInternetSales --")
fact_internet_sales_df = validate_fk(
    fact_internet_sales_df,
    dim_customer_df,
    'CustomerKey',
    'CustomerKey',
    'FactInternetSales',
    'DimCustomer'
)
fact_internet_sales_df = validate_fk(
    fact_internet_sales_df,
    dim_product_df,
    'ProductKey',
    'ProductKey',
    'FactInternetSales',
    'DimProduct'
)
fact_internet_sales_df = validate_fk(
    fact_internet_sales_df,
    dim_salesterritory_df,
    'SalesTerritoryKey',
    'SalesTerritoryKey',
    'FactInternetSales',
    'DimSalesTerritory'
)

print("\n-- Validating foreign keys in FactResellerSales --")
fact_reseller_sales_df = validate_fk(
    fact_reseller_sales_df,
    dim_reseller_df,
    'ResellerKey',
    'ResellerKey',
    'FactResellerSales',
    'DimReseller'
)
fact_reseller_sales_df = validate_fk(
    fact_reseller_sales_df,
    dim_employee_df,
    'EmployeeKey',
    'EmployeeKey',
    'FactResellerSales',
    'DimEmployee'
)
fact_reseller_sales_df = validate_fk(
    fact_reseller_sales_df,
    dim_product_df,
    'ProductKey',
    'ProductKey',
    'FactResellerSales',
    'DimProduct'
```

```
    )
    fact_reseller_sales_df = validate_fk(
        fact_reseller_sales_df,
        dim_salesterritory_df,
        'SalesTerritoryKey',
        'SalesTerritoryKey',
        'FactResellerSales',
        'DimSalesTerritory'
    )
```

4. Loading the Transformed Data to MySQL

After completing the transformations and quality checks, I loaded the cleaned data into a MySQL database. The database is structured with six dimension tables and two fact tables, forming a star schema optimized for reporting and analytics.

Code Snippet: Loading Data to MySQL

```
print("\n-- Loading data into MySQL --")
dim_customer_df.to_sql('dimcustomer', con=engine, if_exists='replace', index=False)
dim_employee_df.to_sql('dimemployee', con=engine, if_exists='replace', index=False)
dim_geography_df.to_sql('dimgeography', con=engine, if_exists='replace', index=False)
dim_product_df.to_sql('dimproduct', con=engine, if_exists='replace', index=False)
dim_reseller_df.to_sql('dimreseller', con=engine, if_exists='replace', index=False)
dim_salesterritory_df.to_sql('dimsalesterritory', con=engine, if_exists='replace',
index=False)

fact_internet_sales_df.to_sql('fact_internetsales', con=engine, if_exists='replace',
index=False)
fact_reseller_sales_df.to_sql('fact_resellersales', con=engine, if_exists='replace',
index=False)
```

Output

Loading dimension tables...
Loading fact tables...
Removed 'CarrierTrackingNumber' column from FactInternetSales.

-- Checking PK uniqueness in dimension tables --
All rows in DimCustomer have a unique 'CustomerKey'.
All rows in DimEmployee have a unique 'EmployeeKey'.
All rows in DimGeography have a unique 'GeographyKey'.
All rows in DimProduct have a unique 'ProductKey'.
All rows in DimReseller have a unique 'ResellerKey'.
All rows in DimSalesTerritory have a unique 'SalesTerritoryKey'.
Filling 56 missing 'Color' cells in DimProduct with 'NA'.
Dropped 1 row(s) from DimSalesTerritory due to missing data.
Dropped 44 row(s) from FactResellerSales due to missing data.

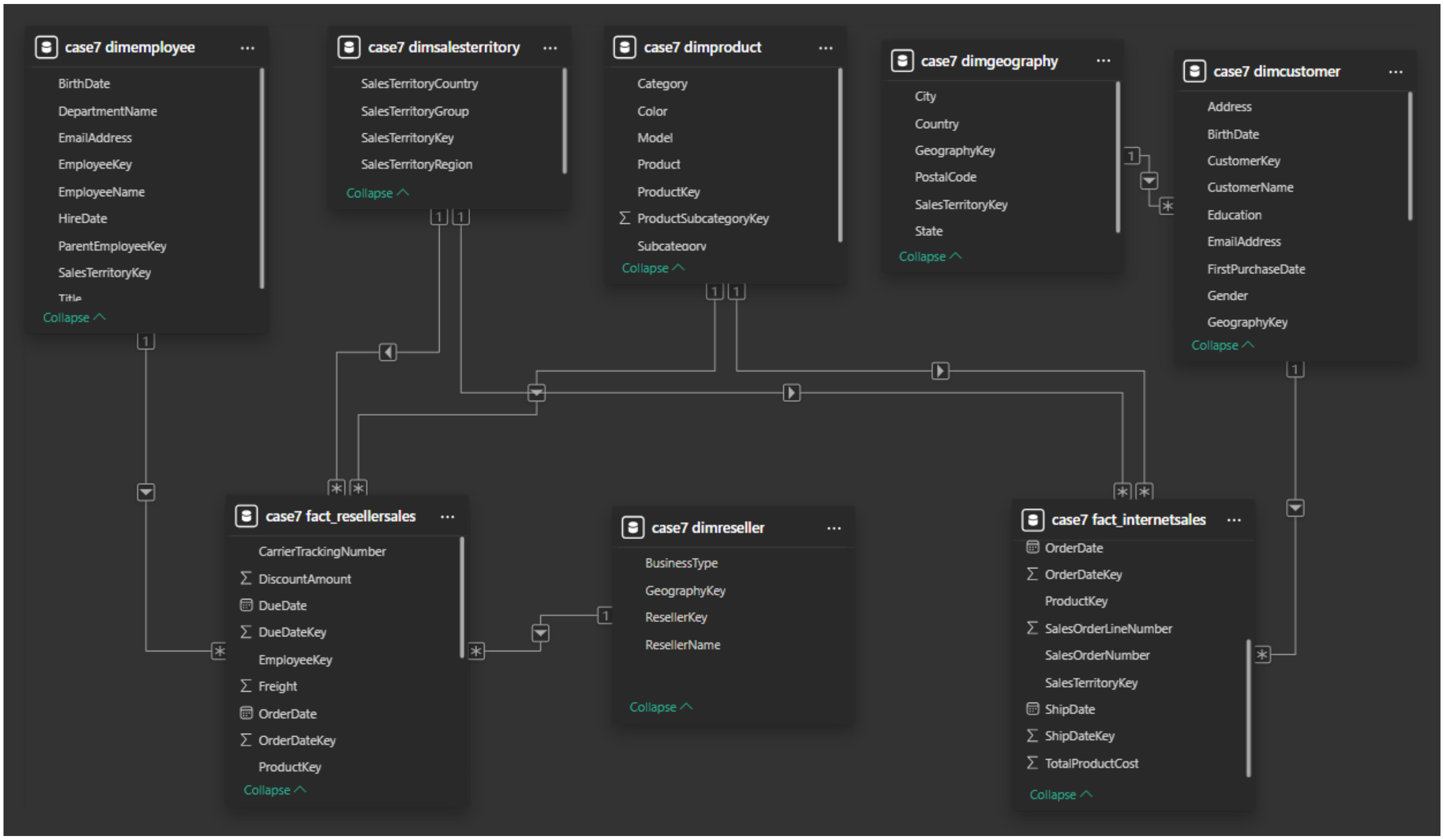
-- Validating foreign keys in FactInternetSales --
All rows in FactInternetSales have a valid foreign key 'CustomerKey'.
All rows in FactInternetSales have a valid foreign key 'ProductKey'.
All rows in FactInternetSales have a valid foreign key 'SalesTerritoryKey'.

-- Validating foreign keys in FactResellerSales --

All rows in FactResellerSales have a valid foreign key 'ResellerKey'.
All rows in FactResellerSales have a valid foreign key 'EmployeeKey'.
All rows in FactResellerSales have a valid foreign key 'ProductKey'.
All rows in FactResellerSales have a valid foreign key 'SalesTerritoryKey'.
...
-- Loading data into MySQL --

ETL process completed successfully!

Schema



5. Analysis Using SQL

Once the data is in MySQL, I can run SQL queries to perform further analysis. For example, here are some SQL commands I used to analyze the data:

Total Internet Sales by Product Category

```
SELECT p.Category AS ProductCategory,
       SUM(f.SalesAmount) AS TotalSales
  FROM fact_internetsales f
  JOIN dimproduct p ON f.ProductKey = p.ProductKey
 GROUP BY p.Category
 ORDER BY TotalSales DESC;
```

ProductCategory	TotalSales
Bikes	28318144.650694
Accessories	700759.9599997756
Clothing	339772.609999959

Top 5 Resellers by Total Sales

```
SELECT r.ResellerName,
       SUM(f.SalesAmount) AS TotalSales
  FROM fact_resellersales f
  JOIN dimreseller r ON f.ResellerKey = r.ResellerKey
 GROUP BY r.ResellerName
 ORDER BY TotalSales DESC
 LIMIT 5;
```

ResellerName	TotalSales
Brakes and Gears	877107.1922999992
Excellent Riding Supplies	853849.1794999994
Vigorous Exercise Company	841908.7707999991
Totes & Baskets Company	816755.5762999995
Retail Mall	799277.8952999999

Total Sales by Territory (Internet Sales)

```
SELECT t.SalesTerritoryRegion AS TerritoryRegion,
       t.SalesTerritoryCountry AS TerritoryCountry,
       SUM(f.SalesAmount) AS TotalSales
  FROM fact_internetsales AS f
  JOIN dimsalesterritory AS t ON f.SalesTerritoryKey = t.SalesTerritoryKey
 GROUP BY t.SalesTerritoryRegion, t.SalesTerritoryCountry
 ORDER BY TotalSales DESC;
```

TerritoryRegion	TerritoryCountry	TotalSales
Australia	Australia	9061000.584401844
Southwest	United States	5718150.812202061
Northwest	United States	3649866.5512009948
United Kingdom	United Kingdom	3391712.2109007137
Germany	Germany	2894312.3382004136
France	France	2644017.7143003284
Canada	Canada	1977844.8620999753
Southeast	United States	12238.849599999994
Northeast	United States	6532.468199999996
Central	United States	3000.829599999997

Reseller Sales by Employee

```
SELECT e.EmployeeName,
       SUM(f.SalesAmount) AS TotalSales
  FROM fact_resellersales AS f
  JOIN dimemployee AS e ON f.EmployeeKey = e.EmployeeKey
 GROUP BY e.EmployeeName
 ORDER BY TotalSales DESC;
```

EmployeeName	TotalSales
Linda Mitchell	10367007.428600065
Jillian Carson	10065803.54290009
Michael Blythe	9293903.005499939
Jae Pak	8503338.647199878
Tsvi Reiter	7171012.751399911
Shu Ito	6427005.555599952
José Saraiva	5926418.357399935
Ranjit Varkey Chudukatil	4509888.932999974
David Campbell	3729945.3500999943
Garrett Vargas	3609447.216299994
Pamela Ansman-Wolfe	3325102.5951999957
Tete Mensa-Annan	2312545.690499986
Rachel Valdez	1790640.2310999965
Lynn Tsoflias	1421810.9251999955
Stephen Jiang	1092123.8561999982
Amy Alberts	732078.4445999986
Syed Abbas	172524.45150000026

Top 5 Resellers by Total Sales

```
SELECT r.ResellerName,
       SUM(f.SalesAmount) AS TotalSales
  FROM fact_resellersales AS f
  JOIN dimreseller AS r ON f.ResellerKey = r.ResellerKey
 GROUP BY r.ResellerName
 ORDER BY TotalSales DESC
 LIMIT 5;
```

ResellerName	TotalSales
Brakes and Gears	877107.1922999992
Excellent Riding Supplies	853849.1794999994
Vigorous Exercise Company	841908.7707999991
Totes & Baskets Company	816755.5762999995
Retail Mall	799277.8952999999

Yearly Internet Sales Trend

```
SELECT YEAR(OrderDate) AS OrderYear,
       SUM(SalesAmount) AS TotalSales
  FROM fact_internetsales
 GROUP BY YEAR(OrderDate)
 ORDER BY OrderYear;
```

OrderYear	TotalSales
2010	43421.03639999999
2011	7075525.9290997805
2012	5842485.195200018
2013	16351550.34000691
2014	45694.7200000001

Top 5 Resellers by Total Sales

```sql
SELECT r.ResellerName,
       SUM(f.SalesAmount) AS TotalSales
  FROM fact_resellersales AS f
  JOIN dimreseller AS r ON f.ResellerKey = r.ResellerKey
 GROUP BY r.ResellerName
 ORDER BY TotalSales DESC
 LIMIT 5;
```

ResellerName	TotalSales
Brakes and Gears	877107.1922999992
Excellent Riding Supplies	853849.1794999994
Vigorous Exercise Company	841908.7707999991
Totes & Baskets Company	816755.5762999995
Retail Mall	799277.8952999999

Internet Sales by Customer Occupation

```sql
SELECT c.Occupation,
       SUM(f.SalesAmount) AS TotalSales
  FROM fact_internetsales AS f
  JOIN dimcustomer AS c ON f.CustomerKey = c.CustomerKey
 GROUP BY c.Occupation
 ORDER BY TotalSales DESC;
```

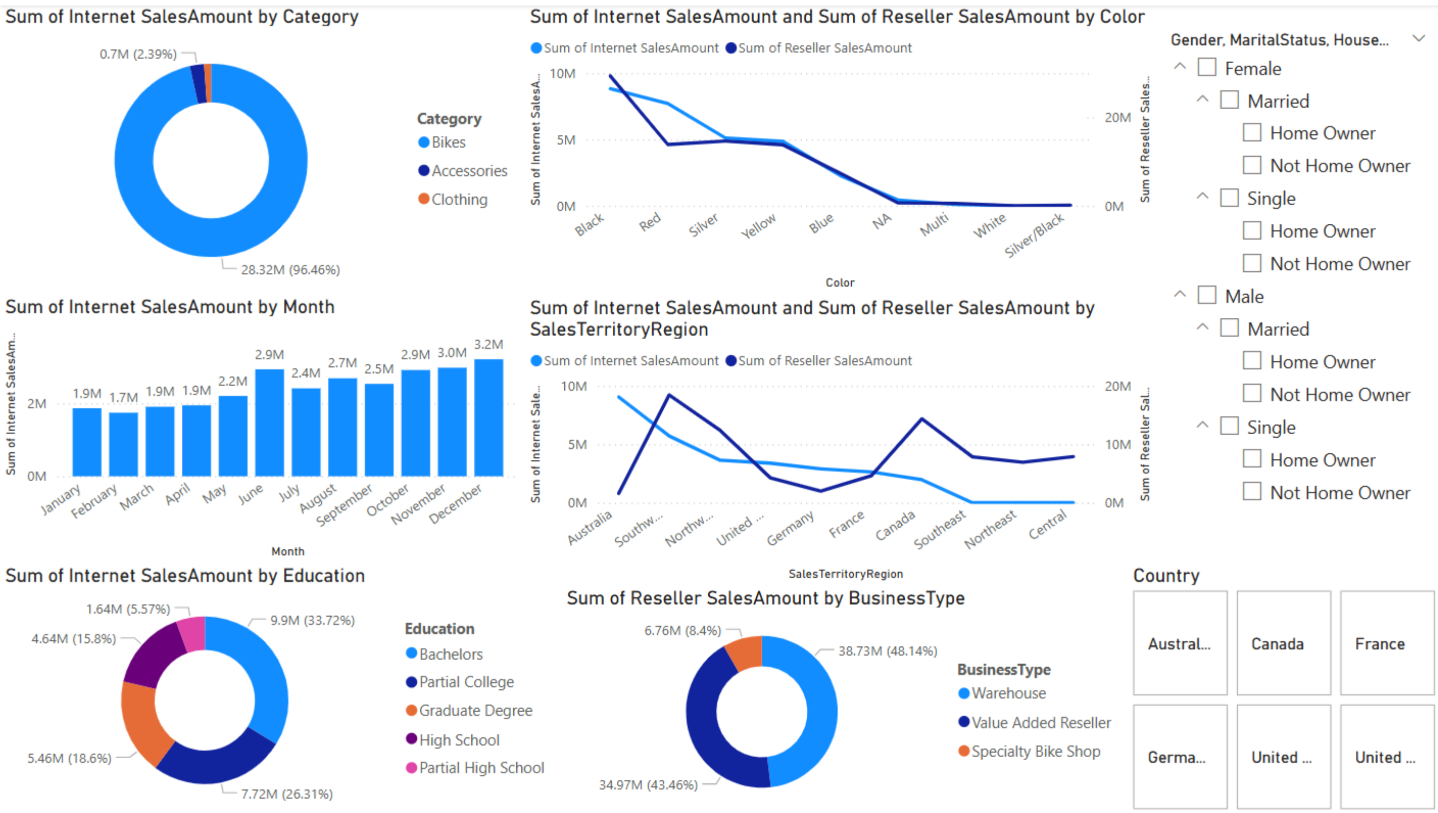
Occupation	TotalSales
Professional	9907977.281102452
Skilled Manual	6440080.858901548
Management	5467861.544501133
Clerical	4684786.643001095
Manual	2857970.893200506

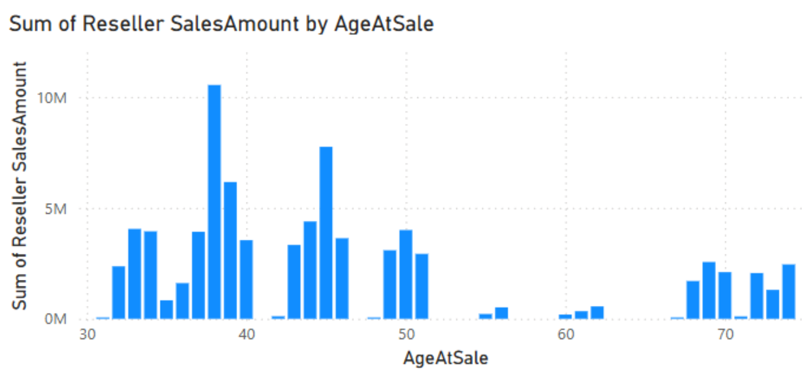
Compare Discount vs. Sales (Reseller Channel)

```sql
SELECT SUM(DiscountAmount) AS TotalDiscount,
       SUM(SalesAmount) AS TotalSales
  FROM fact_resellersales;
```

TotalDiscount	TotalSales
527507.9261999947	80450596.98229924

6. KPI Tracking & Monitoring Results





Conclusion

Bikes are the top-selling product category.

June is a peak month for both Internet and Reseller sales.

Age 30–50 is the most active buying demographic.

Higher-educated customers (Bachelor's, Graduate, Partial College) account for a large share of purchases.

Black, Red, Silver colors are most popular.