

Data Engineering Assessment

Task Definition –

- a) By make use of the **invoices.xls** data file as your source, create a Datawarehouse model with best practices in place, using Kimball methodology.*
- b) Populate the model created in step **a)** with the data provided in the excel sheet.*
- c) Analyze any abnormalities (if any) in the data provided and take any action needed (where possible).*
- d) List any assumptions taken into consideration.*
- e) Provide 3 different aggregations one might use for reporting purposes.*

Question a

The purpose of this task is to design a **Data Warehouse (DW)** based on the transactional data from the provided invoices.xls file. The design follows the **Kimball methodology**, using dimensional modeling to enable efficient and insightful Business Intelligence (BI) reporting.

This approach focuses on delivering a **Star Schema**, separating the data into **fact** and **dimension tables**, aligned with best practices from the Kimball methodology.

Based on the nature of the data (sales invoices), the core requirements are based on the following items:

- Track sales by **product, customer, country, date**, and **time of day**.
- Identify seasonality and sales trends over time.
- Analyze top-performing products and customers.
- Enable geographic reporting across **countries** and **continents**.

Dimensional Model – Star Schema

The ETL pipeline first loads raw data into a temporary staging table, FactSales_Staging, which serves as a landing zone for cleaned and enriched transactional data. This layer includes raw transformations and allows validation, auditing, and surrogate key resolution before final insertion into the Fact Table.

Once staging is complete, the data is joined with the dimension tables to resolve surrogate keys, and then loaded into the central fact table: FactSales.

The dimensional model includes:

- Fact Table

- FactSales: Measures and foreign keys to dimensional attributes
- Staging Table
 - FactSales_Staging: Landing zone for cleaned and enriched transactional data.
- Dimension Tables
 - DimCustomer: Customer demographics and geography
 - DimProduct: Product metadata, categories, seasonality
 - DimInvoice: Transaction-level flags (returns, discounts)
 - DimDate: Calendar attributes
 - DimTime: Time-of-day breakdowns

DimInvoice

Captures metadata about invoices.

Column	Description
invoice_pk	Surrogate key
invoice_id	Natural invoice ID
returned	Y/N, based on the Invoice ID like 'C%'
discount	Y/N, based on " Quantity " < 0 and " StockCode " != 'D'

DimCustomer

Stores information about customers.

Column	Description
customer_pk	Surrogate key
customer_id	Natural customer ID
country	Customer's country
continent	New extracted column for geographical analysis

DimProduct

Represents the products being sold.

Column	Description
product_pk	Surrogate key
product_id	Product/business ID
description	Product description
product_category	Optional categorization
seasonal (Y/N)	Flag indicating if a product is seasonal (e.g., based on keywords)

DimDate

Captures calendar-related components of the invoice date.

Column	Description
date_pk	Surrogate key
date	Full date, in format 'YYYYMMDD'
year, month, day, weekday, quarter	Date components for time-based analysis

DimTime

Captures the time portion of the invoice datetime.

Column	Description
time_pk	Surrogate key
time	Time in 'HHMMSS'
hour, minute, second	Time components
time_of_day	Extracted column (Morning / Afternoon / Evening classification)

Staging Table

FactSales_Staging

Column	Description
invoice_id	Business identifier for the transaction

product_id	Business identifier for the product sold
customer_id	Business identifier for the customer making the purchase
date_id	Date in YYYYMMDD format extracted from the invoice timestamp
time_id	Time in HHMMSS format extracted from the invoice timestamp
quantity	Number of product units purchased
price	Price per unit of product at time of sale
total_amount	Extracted column (price * quantity)

Facts Table

FactSale table

Column	Description
invoice_fk	Surrogate key referencing DimInvoice.invoice_pk
product_fk	Surrogate key referencing DimProduct.product_pk
customer_fk	Surrogate key referencing DimCustomer.customer_pk
date_fk	Surrogate key referencing DimDate.date_pk
time_fk	Surrogate key referencing DimTime.time_pk
quantity	Number of product units sold (or returned if negative)
price	Unit price of the product at the time of sale
total_amount	Total transaction value (calculated as quantity * price)

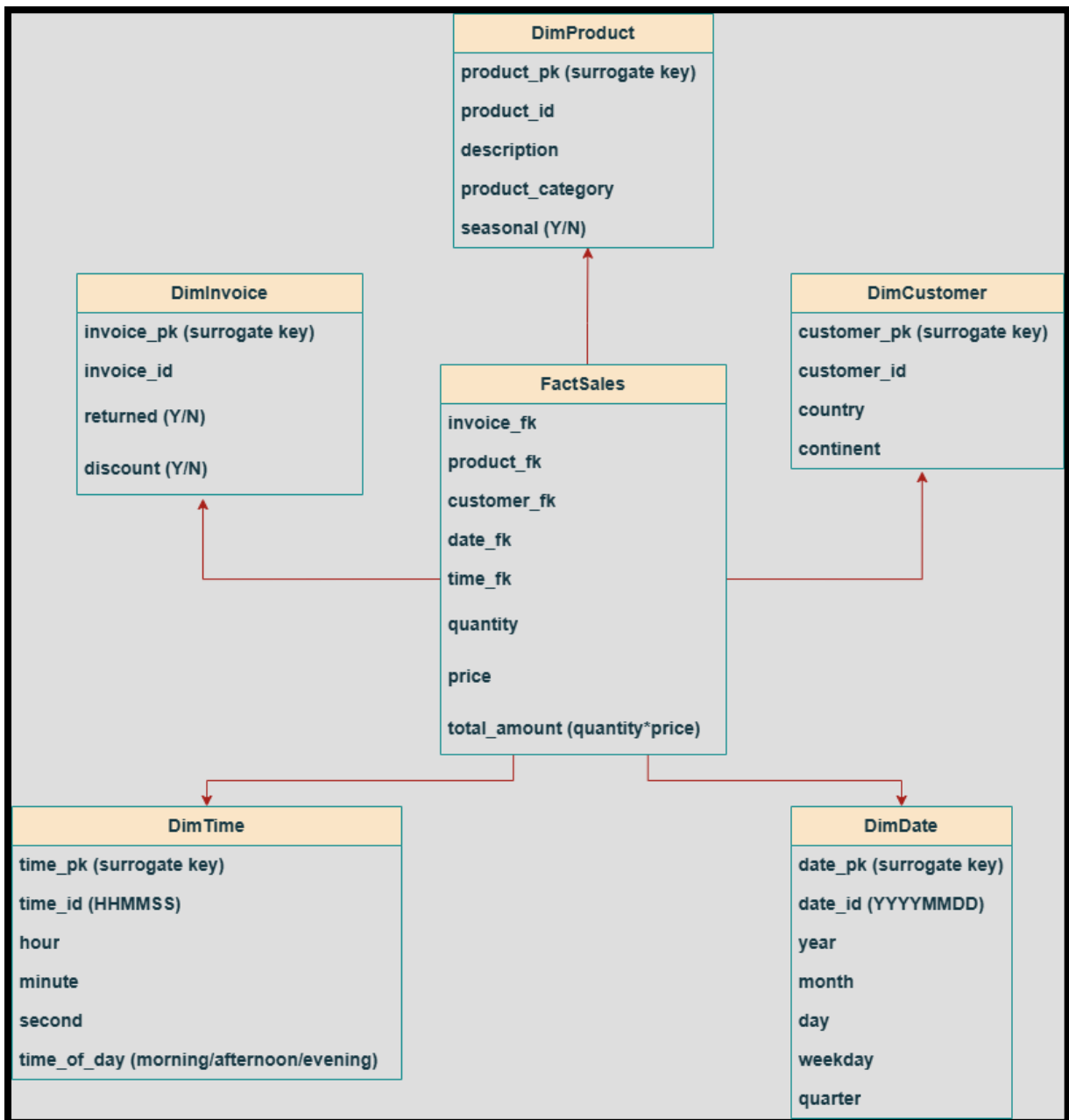


Image 1: Data Warehouse model based on Kimball methodology.

In the file with name **create_statements.sql**, you can find the create statements of the Dimension, Staging and Fact tables as long as for the project needs I have created a relevant database with name **SalesDataWarehouse**. All dimension tables use surrogate keys as primary keys and foreign keys in the fact table. In addition to the table creation script in `create_statements.sql`, non-clustered indexes were implemented on the natural keys of each Dimension table: *DimInvoice.invoice_id*, *DimCustomer.customer_id*, *DimProduct.product_id*, *DimDate.date_id*, and *DimTime.time_id*. These indexes were added to optimize query performance, particularly for JOIN operations during the ETL process in order to enhance the efficiency of reporting queries that filter or group by these fields, ensuring scalability and fast data retrieval as the dataset grows.

The proposed dimensional model is fully aligned with the principles of the Kimball methodology, offering a robust foundation for analytical reporting. All dimension tables utilize surrogate keys, which serve as foreign keys in the central fact table (FactSales), ensuring optimal performance. While surrogate keys are used for joins, natural or business keys—such as CustomerID and StockCode are retained within the dimensions for reference and traceability. The model is intentionally denormalized to enhance usability and support fast, intuitive querying by business users. The inclusion of both DimDate and DimTime enables granular temporal analysis, facilitating both historical trends and intra-day performance evaluation. The FactSales table has been designed at the grain of a single product per invoice line, capturing all relevant measurable metrics such as quantity, unit price, and total sales amount.

This model supports a wide range of analytical use cases, including identifying top-selling products by customer or country, analyzing sales performance across weekdays and quarters, seasonal trends, and determining peak sales hours. Ultimately, this architecture provides clarity, efficiency, scalability, and flexibility—delivering a comprehensive and extensible platform for data-driven decision-making.

Question b

In the file with name **etl.ipynb**, you can find the etl pipeline which I followed in order to to ensure high data quality, analytical flexibility, and consistency with dimensional modeling best practices. The process is modular, driven by reusable ETL functions implemented in Python and consisted of multiple stages before loading the data into the target MSSQL Server database:

1. Extract
 - Load the data from the given csv file with UTF-8 encoding.
2. Transform
 - Dimensions tables: Fields creation for every Dimension tables (i.e. continent field).
 - Data Cleaning: Excluding rows with null values on Country or Customer ID.
 - Loaded dimension tables using drop_duplicates on their business/natural keys.
 - Contextual Enrichment.
 - Product Category Enrichment.
3. Load
 - Loading the transformed or extracted data into Dimension and FactsSales_staging tables.
4. Fact Table Staging & Surrogate Mapping
 - Created a FactSales_Staging table using natural keys.
 - Mapped all foreign key references using JOIN to the dimension tables.
 - Final insert into FactSales uses surrogate keys only, in line with Kimball methodology.

The ETL pipeline implemented in the etl.ipynb file, was designed to perform a **full load** of the provided csv file (dataset) into SalesDataWarehouse database. A delta approach which would handle incremental updates by processing only new or modified records, was not

implemented in this phase but could be considered as a future enhancement to support data integration and scalability in a production environment.

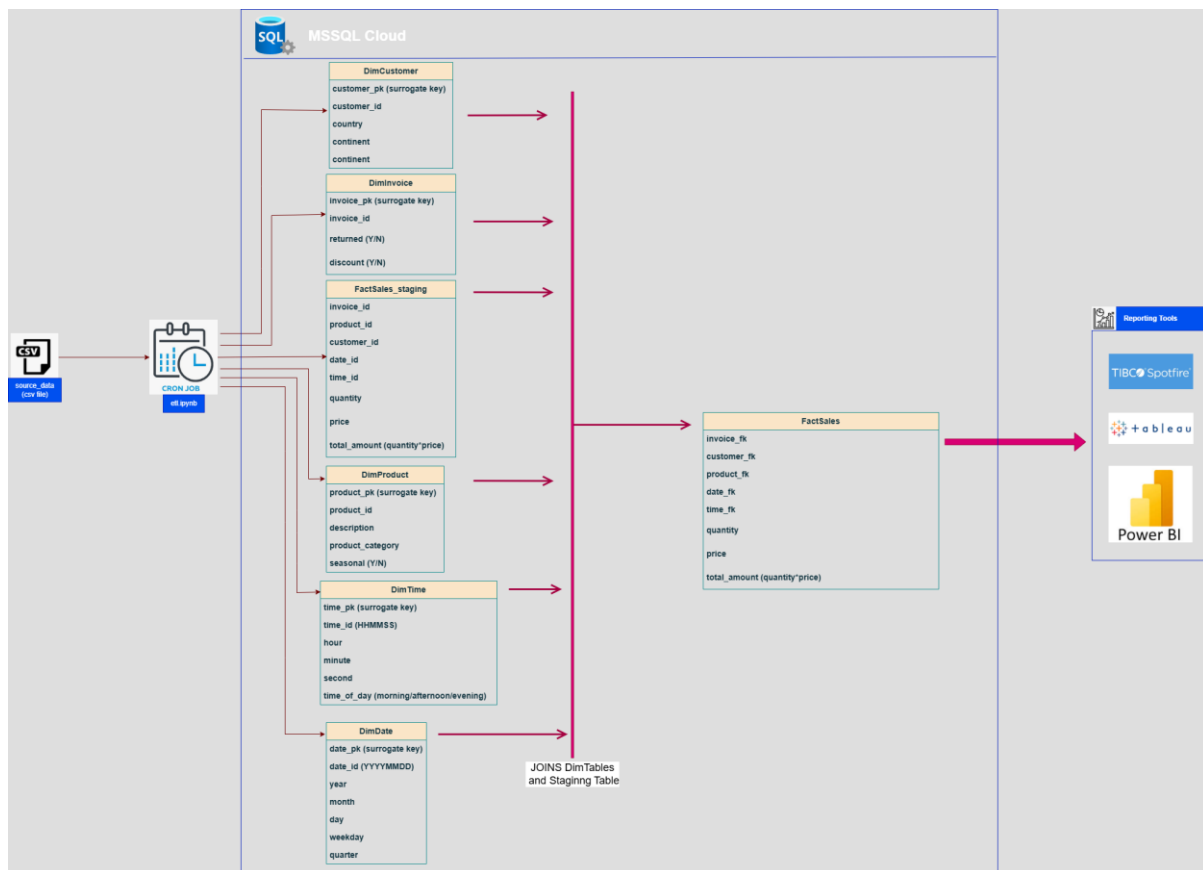


Image 2: End-to-End Data Warehouse Architecture.

Question c

During the data profiling and ETL implementation process, several abnormalities and inconsistencies were identified in the dataset. Each issue was handled through targeted transformations to ensure data quality, modeling accuracy, and readiness for analytics and will be analyzed in the below areas:

1. **Missing values**

The columns Customer_ID, Country and Prices contain null values:

- Records with null Price or Price < 0 were excluded, as price is critical for metric calculation.
- Records with null Customer ID were retained using a default customer surrogate (-1).
- Null values in Country were replaced with 'Unknown'.

2. **Flag Columns**

A transaction was flagged as a **return** if:

- The quantity was negative **and** the invoice_id string started with "C" (assumed to be the notation for credit notes).

A row was considered a **discount line** if:

- Quantity < 0 and StockCode == 'D' .

3. **Customer Info**

Country column was used to enrich customer geography, and a mapping to continent was applied using a pre-defined dictionary. Countries not explicitly mapped were assigned with value "Unknown".

4. **Time and Date**

The InvoiceDate column was assumed to contain extract the following time-based attributes:

- date_id, year, month, day, weekday, quarter
- time_id, hours, minutes, seconds, and time_of_day classification

5. **Product Categorization and Seasonal Fields**

Products were categorized using a keyword-based scoring approach from the Description field. Each product was assigned to the category with the highest keyword match score using a list of domain-relevant terms.

Products were flagged as seasonal (Y) if their description is related to holidays or seasonal events (e.g. "Christmas", "Easter", "Valentine").

Descriptions with no matches were labeled as "Unknown" in the category field and "N" in the seasonal flag.

6. **Trimming**

All columns were stripped of whitespace to void false mismatches during joins.

7. **Duplicate handling**

Use of drop_duplicates in order to avoid duplication in Dimension tables.

8. **Handling zero-priced records**

Records with price equals to zero, excluded because they do not reflect any actual transaction.

Last but not least, surrogate keys were used in all dimension tables to support referential integrity, performance, and historical flexibility. Natural keys (e.g. Customer ID, Invoice Number) were preserved for reference purposes only.

Question d

Below you will find which assumptions taken into consideration:

Country and Geography

- Null values in the Country column were replaced with "Unknown".
- A predefined mapping from country to continent was used; unmatched countries defaulted to "Unknown".

Invoice Info

- An invoice_id starting with "C" was interpreted as a **credit note**, and rows with negative quantity were flagged as **returns**.

Discount Logic

- Rows with StockCode = 'D' and negative quantity were treated as **discount lines**, not product returns.

Invalid and Incomplete Records

- Records with missing InvoiceDate or Price were excluded.
- Transactions with Price ≤ 0 were also excluded, as they do not represent meaningful business activity.

Product Categorization

- Product categories were inferred using keyword-based logic applied to the Description field.
- If multiple matches occurred, the category with the **most keyword hits** was assigned.
- Descriptions with no matches were labeled as "Unknown".

Seasonal Classification

- Products were marked as seasonal if their description contained predefined seasonal terms (e.g., "Christmas", "Valentine", "Easter").

Time and Date Derivation

- InvoiceDate was assumed to be in local time and used to derive all related temporal fields:
 - date_id, year, month, day, weekday, quarter, time_id, hour, minute, second, and time_of_day.

String Cleanup

- All string fields (e.g., Description, Country, StockCode) were normalized via trimming and lowercasing to ensure consistency in joins and transformations.

Data Model Design

- All dimension tables use **surrogate keys** as primary keys.
- Natural keys from the source (e.g., Invoice, StockCode) are preserved for traceability but not used in joins.

Question e

I have created three representative use cases demonstrating Data Warehouse's ability to generate business insights through SQL queries on the star schema

Query 1: Top 10 Customers by Purchase Volume

SELECT TOP 10

c.customer_id,

SUM(f.quantity) AS total_quantity,

c.country,

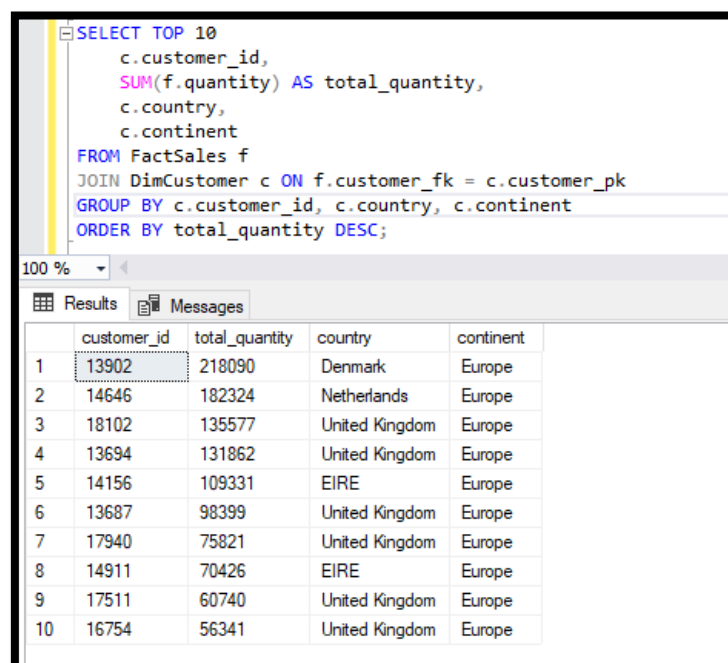
c.continent

FROM FactSales f

JOIN DimCustomer c ON f.customer_fk = c.customer_pk

GROUP BY c.customer_id, c.country, c.continent

ORDER BY total_quantity DESC;



	customer_id	total_quantity	country	continent
1	13902	218090	Denmark	Europe
2	14646	182324	Netherlands	Europe
3	18102	135577	United Kingdom	Europe
4	13694	131862	United Kingdom	Europe
5	14156	109331	EIRE	Europe
6	13687	98399	United Kingdom	Europe
7	17940	75821	United Kingdom	Europe
8	14911	70426	EIRE	Europe
9	17511	60740	United Kingdom	Europe
10	16754	56341	United Kingdom	Europe

Image 3: SQL results of query 1.

2. Total Sales per Month and Product Category

SELECT

d.year,

d.month,

p.product_category,

SUM(f.total_amount) AS total_sales

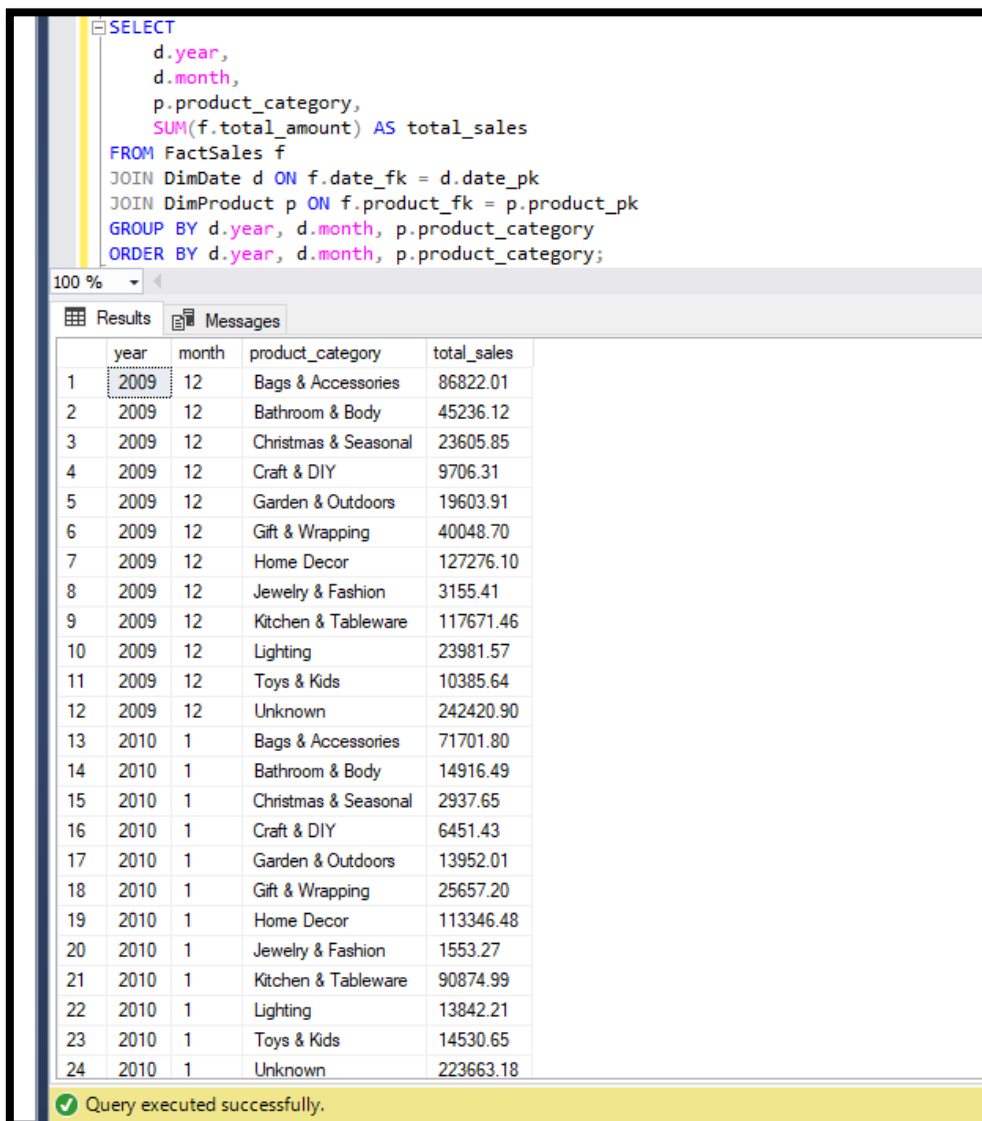
FROM FactSales f

JOIN DimDate d ON f.date_fk = d.date_pk

JOIN DimProduct p ON f.product_fk = p.product_pk

GROUP BY d.year, d.month, p.product_category

ORDER BY d.year, d.month, p.product_category;



```
SELECT
    d.year,
    d.month,
    p.product_category,
    SUM(f.total_amount) AS total_sales
FROM FactSales f
JOIN DimDate d ON f.date_fk = d.date_pk
JOIN DimProduct p ON f.product_fk = p.product_pk
GROUP BY d.year, d.month, p.product_category
ORDER BY d.year, d.month, p.product_category;
```

	year	month	product_category	total_sales
1	2009	12	Bags & Accessories	86822.01
2	2009	12	Bathroom & Body	45236.12
3	2009	12	Christmas & Seasonal	23605.85
4	2009	12	Craft & DIY	9706.31
5	2009	12	Garden & Outdoors	19603.91
6	2009	12	Gift & Wrapping	40048.70
7	2009	12	Home Decor	127276.10
8	2009	12	Jewelry & Fashion	3155.41
9	2009	12	Kitchen & Tableware	117671.46
10	2009	12	Lighting	23981.57
11	2009	12	Toys & Kids	10385.64
12	2009	12	Unknown	242420.90
13	2010	1	Bags & Accessories	71701.80
14	2010	1	Bathroom & Body	14916.49
15	2010	1	Christmas & Seasonal	2937.65
16	2010	1	Craft & DIY	6451.43
17	2010	1	Garden & Outdoors	13952.01
18	2010	1	Gift & Wrapping	25657.20
19	2010	1	Home Decor	113346.48
20	2010	1	Jewelry & Fashion	1553.27
21	2010	1	Kitchen & Tableware	90874.99
22	2010	1	Lighting	13842.21
23	2010	1	Toys & Kids	14530.65
24	2010	1	Unknown	223663.18

Query executed successfully.

Image 4: SQL results of query 2.

Query 3: Return Rate by Country

SELECT

c.country,

*CAST(SUM(CASE WHEN i.returned = 'Y' THEN 1 ELSE 0 END) * 100.0 / COUNT(*) AS DECIMAL(5,2)) AS return_rate_percent*

FROM FactSales f

JOIN DimInvoice i ON f.invoice_fk = i.invoice_pk

JOIN DimCustomer c ON f.customer_fk = c.customer_pk

GROUP BY c.country

ORDER BY return_rate_percent DESC;

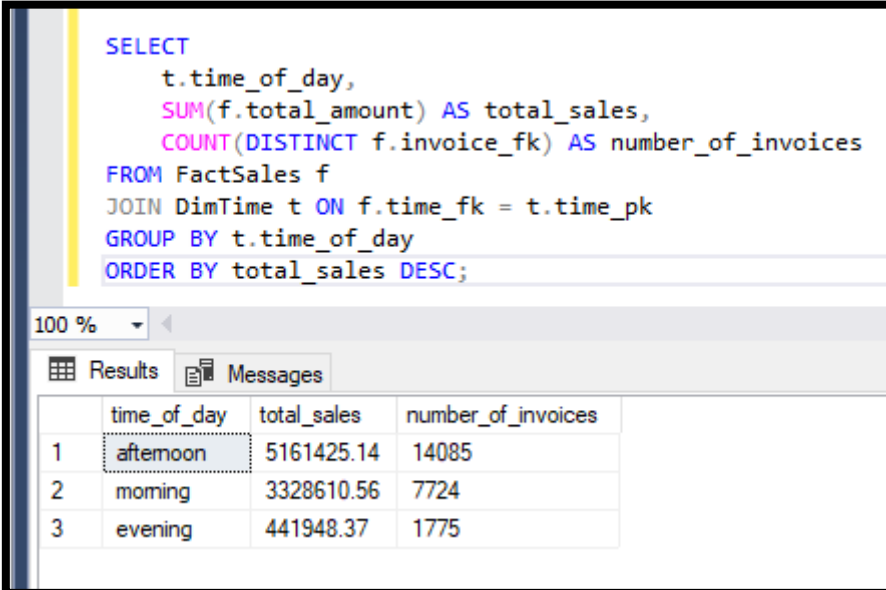
100 %

Results Messages

	country	return_rate_percent
1	Japan	26.75
2	Korea	15.94
3	Channel Islands	9.15
4	Poland	5.85
5	Germany	5.74
6	USA	5.53
7	France	4.28
8	Portugal	3.94
9	Australia	3.82
10	Spain	3.48
11	Italy	2.88
12	Cyprus	2.59
13	Austria	2.47
14	EIRE	2.41
15	United Kingdom	2.30
16	Finland	2.30
17	Malta	2.14
18	Norway	2.09
19	Denmark	2.09
20	Sweden	2.06
21	Belgium	1.69
22	Unspecified	1.64
23	Netherlands	1.48
24	Switzerland	1.38
25	Greece	0.93
26	United Arab E...	0.91

Image 5: SQL results of query 3.

Query 4: Total Sales and Invoice Count by Time of Day



```
SELECT
    t.time_of_day,
    SUM(f.total_amount) AS total_sales,
    COUNT(DISTINCT f.invoice_fk) AS number_of_invoices
FROM FactSales f
JOIN DimTime t ON f.time_fk = t.time_pk
GROUP BY t.time_of_day
ORDER BY total_sales DESC;
```

100 %

Results Messages

	time_of_day	total_sales	number_of_invoices
1	afternoon	5161425.14	14085
2	morning	3328610.56	7724
3	evening	441948.37	1775

Image 6: SQL results of query 4.