Assignment 2
Data Warehouse Project

Vedant Marwadi - 23208466

Data Warehouse and Big Data,
Master of Computer and Information Sciences,
Auckland University of Technology

Contents

Section A

1.1. Overview of the Project

1.1.1 Introduction

As part of the initiative to gain a deeper understanding of customer shopping behavior and refine business strategies for OfficeProducts, a comprehensive data warehouse is developed. This report delves into the design process behind the snowflake schema that underpins the data warehouse, highlighting the purpose and relationships of the fact and dimension tables. It also explains how these elements work together to support diverse business queries and facilitate in-depth performance evaluations, ensuring that the company can effectively respond to changing market demands and make informed, strategic choices.

1.1.2 Business Context

OfficeProducts is a leading supplier of office products in New Zealand, with a large portion of its sales conducted through its online platform. Understanding client shopping behavior is crucial for them in order to optimize sales strategies, effectively manage product promotions, and control logistics, such as reducing courier costs. By implementing a well-designed data warehouse, OfficeProducts can efficiently analyze transaction data, gain valuable insights, and support data-driven decision-making to enhance business operations.

1.1.3 Objective of the Data Warehouse

The primary objective of the data warehouse is to allow OfficeProducts to:

- Analyze sales performance by product, client, channel, and time period.
- Understand product-specific trends and determine best-selling products.
- Optimize promotions based on product category, client segment, and time-based factors.
- Aggregate sales and other metrics to help with forecasting, pricing strategy, and marketing campaigns.

To support these objectives, the data warehouse needs to accommodate multiple levels of aggregation across various business dimensions. Therefore, the snowflake schema is chosen to normalize the data and enable detailed, cross-dimensional analysis while avoiding data redundancy.

1.2. Snowflake Schema Design

1.2.1 Snowflake Schema Architecture

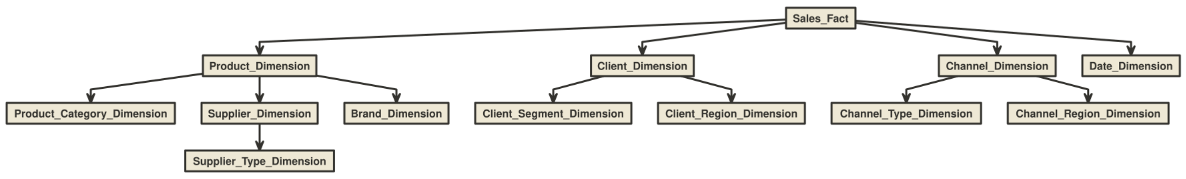


Figure 1.1: Snowflake Schema Architecture for the OfficeProducts Firm

1.2.2 Entity-Relationship Diagram

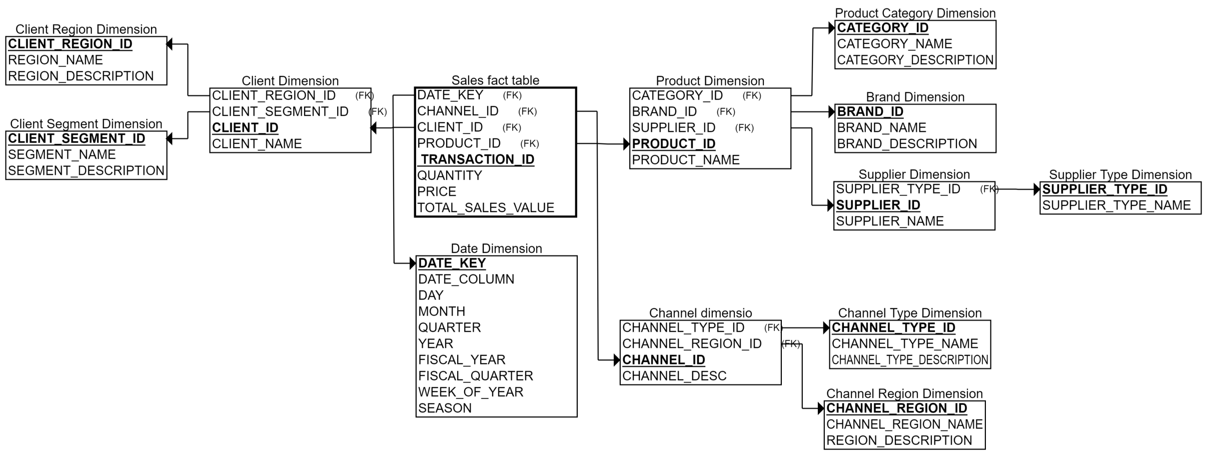


Figure 1.2: ER Diagram of the Snowflake Schema for the OfficeProducts Firm

1.2.3 Explanation

In this snowflake schema, the central table, referred to as `Sales_Fact`, serves as the primary repository for storing quantitative data associated with various sales transactions of the OfficeProducts firm. The `Sales_Fact` table aggregates key performance indicators to facilitate thorough analysis. It is linked to four dimension tables, each of which offers a unique set of descriptive attributes that help contextualize the numerical data within the fact table.

The `Product_Dimension` is one of the primary dimensions linked to the `Sales_Fact` table. It is further broken down into three sub-dimensions: `Product_Category_Dimension`, `Supplier_Dimension`, and `Brand_Dimension`, providing more detailed information about the products being sold. Additionally, the `Supplier_Dimension` links to the `Supplier_Type_Dimension`, enabling a more granular categorization of supplier types. This hierarchical structure helps maintain the integrity of product-related data and minimizes data redundancy.

Similarly, the `Client_Dimension` is decomposed into `Client_Segment_Dimension` and `Client_Region_Dimension`, offering insights into the segments and regions to which clients belong. This setup supports the analysis of sales data based on client demographics and geographic information. Another dimension, the `Channel_Dimension`, splits into `Channel_Type_Dimension` and `Channel_Region_Dimension`, facilitating a detailed understanding of sales distribution by channel type and regional performance of these channels.

Lastly, the `Date_Dimension` provides temporal context, such as year, quarter, month, and day, to analyze sales trends over time. This dimension is directly connected to the `Sales_Fact` table without any further subdivisions, as date-related information generally does not require additional normalization.

1.3. Fact Table

1.3.1 Sales_Fact

`Sales_Fact` table includes details about products sold, clients making purchases, the sales channel used, and the transaction date. The table also tracks the number of items sold, the price per unit, and the total sales value for each transaction. It supports a wide range of aggregations, as demonstrated in the Aggregation Possibilities section using the `Product_Dimension`, `Client_Dimension`, `Channel_Dimension`, and `Date_Dimension`.

Attributes of Sales_Fact Table:

Attribute	Details
TRANSACTION_ID (PK)	Unique identifier for each sales transaction.
PRODUCT_ID (FK)	Links the transaction to a specific product from the Product_Dimension.
CLIENT_ID (FK)	Links the transaction to a specific client from the Client_Dimension.
CHANNEL_ID (FK)	Links the transaction to the sales channel from the Channel_Dimension.
DATE (FK)	Captures the date of the transaction from the Date_Dimension.
QUANTITY	The number of products sold in a transaction.
PRICE	The price per unit of the product during the transaction.
TOTAL_SALES_VALUE	The total sales amount calculated as QUANTITY * PRICE.

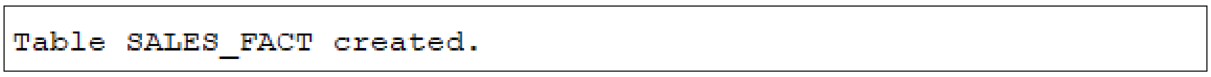
Table 1.1: Description of Attributes in the Sales_Fact Table

SQL Statement to Create the Sales_Fact Table:

```sql
CREATE TABLE Sales_Fact (
    TRANSACTION_ID VARCHAR2(8) PRIMARY KEY,
    PRODUCT_ID VARCHAR2(6),
    CLIENT_ID VARCHAR2(4),
    CHANNEL_ID VARCHAR2(3),
    DATE_KEY NUMBER(8),
    QUANTITY NUMBER(3,0),
    PRICE NUMBER(5,2),
    TOTAL_SALES_VALUE NUMBER(10,2),
    FOREIGN KEY (PRODUCT_ID) REFERENCES Product_Dimension(PRODUCT_ID),
    FOREIGN KEY (CLIENT_ID) REFERENCES Client_Dimension(CLIENT_ID),
    FOREIGN KEY (CHANNEL_ID) REFERENCES Channel_Dimension(CHANNEL_ID),
    FOREIGN KEY (DATE_KEY) REFERENCES Date_Dimension(DATE_KEY)
);
```

Listing 1.1: Query to Set Up Sales_Fact Table

Output:



```
Table SALES_FACT created.
```

Figure 1.3: Output Showing Sales_Fact Table Creation

1.4. Dimension Tables

1.4.1 Product_Dimension

The Product_Dimension table provides detailed information about each product sold by OfficeProducts. It captures attributes such as the product's name, associated category, supplier, and brand. Each product is uniquely identified by a product ID. The table supports advanced filtering and slicing of data through its connections to the `Product_Category_Dimension`, `Supplier_Dimension`, and `Brand_Dimension` tables.

Attributes of Product_Dimension:

Attribute	Details
PRODUCT_ID (PK)	A unique identifier for each product.
PRODUCT_NAME	The name of the product.
CATEGORY_ID (FK)	Links to the category the product belongs to from the Product_Category_Dimension.
SUPPLIER_ID (FK)	Links to the supplier that provides the product from the Supplier_Dimension.
BRAND_ID (FK)	Links to the brand associated with the product from the Brand_Dimension.

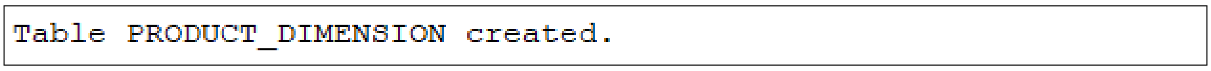
Table 1.2: Description of Attributes in the Product_Dimension Table

SQL Statement to Create the Product_Dimension Table:

```sql
CREATE TABLE Product_Dimension (
    PRODUCT_ID VARCHAR2(6) PRIMARY KEY,
    PRODUCT_NAME VARCHAR2(30),
    CATEGORY_ID VARCHAR2(4),
    SUPPLIER_ID VARCHAR2(5),
    BRAND_ID VARCHAR2(4),
    FOREIGN KEY (CATEGORY_ID) REFERENCES Product_Category_Dimension(CATEGORY_ID),
    FOREIGN KEY (SUPPLIER_ID) REFERENCES Supplier_Dimension(SUPPLIER_ID),
    FOREIGN KEY (BRAND_ID) REFERENCES Brand_Dimension(BRAND_ID)
);
```

Listing 1.2: Query to Set Up Product_Dimension Table

Output:



```
Table PRODUCT_DIMENSION created.
```

Figure 1.4: Output Showing Product_Dimension Table Creation

1.4.2 Client_Dimension

The `Client_Dimension` table holds data about customers making purchases. It provides essential information about clients. Each client is uniquely identified and linked to additional dimensions such as client segments and geographic regions. This structure allows for a deeper understanding of client behavior and performance when analyzed alongside sales data.

Attributes of Client_Dimension:

Attribute	Details
CLIENT_ID (PK)	A unique identifier for each client.
CLIENT_NAME	The name of the client making purchases.
CLIENT_SEGMENT_ID (FK)	Links to the segment the client belongs to from the Client_Segment_Dimension.
CLIENT_REGION_ID (FK)	Links to the geographic region of the client from the Client_Region_Dimension.

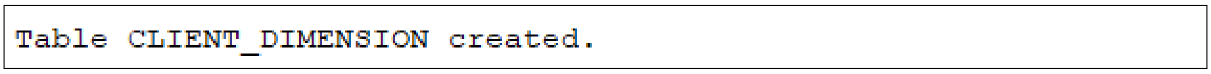
Table 1.3: Description of Attributes in the Client_Dimension Table

SQL Statement to Create the Client_Dimension Table:

```sql
CREATE TABLE Client_Dimension (
    CLIENT_ID VARCHAR2(4) PRIMARY KEY,
    CLIENT_NAME VARCHAR2(30),
    CLIENT_SEGMENT_ID VARCHAR2(3),
    CLIENT_REGION_ID VARCHAR2(3),
    FOREIGN KEY (CLIENT_SEGMENT_ID) REFERENCES Client_Segment_Dimension(
        ↪ CLIENT_SEGMENT_ID),
    FOREIGN KEY (CLIENT_REGION_ID) REFERENCES Client_Region_Dimension(CLIENT_REGION_ID
        ↪ )
);
```

Listing 1.3: Query to Set Up Client_Dimension Table

Output:



```
Table CLIENT_DIMENSION created.
```

Figure 1.5: Output Showing Client_Dimension Table Creation

1.4.3 Channel_Dimension

The `Channel_Dimension` table captures details about the various sales channels used by OfficeProducts. Each channel is uniquely identified and linked to dimensions that define the channel type and its operating region. This structure facilitates detailed analysis of sales data across multiple channels when combined with the `Channel_Type_Dimension` and `Channel_Region_Dimension` tables.

Attributes of Channel_Dimension:

Attribute	**Details**
CHANNEL_ID (PK)	A unique identifier for each sales channel.
CHANNEL_DESC	A description of the channel.
CHANNEL_TYPE_ID (FK)	Links to the type of channel from the Channel_Type_Dimension.
CHANNEL_REGION_ID (FK)	Links to the region the sales channel operates in from the Channel_Region_Dimension.

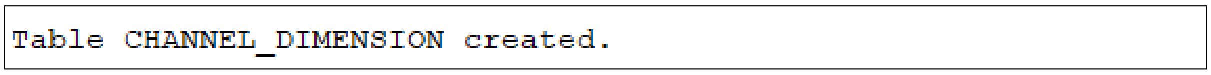
Table 1.4: Description of Attributes in the Channel_Dimension Table

SQL Statement to Create the Channel_Dimension Table:

```sql
CREATE TABLE Channel_Dimension (
    CHANNEL_ID VARCHAR2(3) PRIMARY KEY,
    CHANNEL_DESC VARCHAR2(30),
    CHANNEL_TYPE_ID VARCHAR2(3),
    CHANNEL_REGION_ID VARCHAR2(3),
    FOREIGN KEY (CHANNEL_TYPE_ID) REFERENCES Channel_Type_Dimension(CHANNEL_TYPE_ID),
    FOREIGN KEY (CHANNEL_REGION_ID) REFERENCES Channel_Region_Dimension(
        ↪ CHANNEL_REGION_ID)
);
```

Listing 1.4: Query to Set Up Channel_Dimension Table

Output:



```
Table CHANNEL_DIMENSION created.
```

Figure 1.6: Output Showing Channel_Dimension Table Creation

1.4.4 Date_Dimension

The Date_Dimension table provides extensive details about each calendar date, supporting temporal analysis of sales data. It includes attributes for day, month, quarter, and year, as well as fiscal and seasonal information. This table enables granular time-based reporting and trend analysis, making it a vital component for understanding sales patterns over time. Each date is uniquely identified by a surrogate key.

Attributes of Date_Dimension:

Attribute	Details
DATE_KEY (PK)	A unique surrogate key representing each calendar date.
DATE_COLUMN	The specific calendar date represented in a standard timestamp format.
DAY	Indicates the numerical day of the month for each date.
MONTH	Represents the month of the year in which the transaction occurred.
QUARTER	The quarter of the year.
YEAR	The year in which the transaction occurred.
FISCAL_YEAR	The fiscal year related to the transaction.
FISCAL_QUARTER	The fiscal quarter corresponding to the transaction.
WEEK_OF_YEAR	The week of the year.
SEASON	The season of the year.

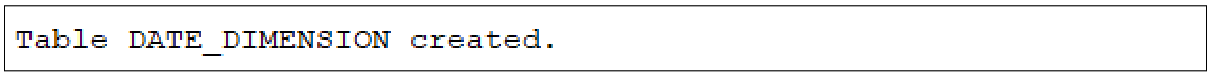
Table 1.5: Description of Attributes in the Date_Dimension Table

SQL Statement to Create the Date_Dimension Table:

```sql
CREATE TABLE Date_Dimension (
    DATE_KEY NUMBER(8) PRIMARY KEY,
    DATE_COLUMN TIMESTAMP,
    DAY NUMBER(2),
    MONTH NUMBER(2),
    QUARTER NUMBER(1),
    YEAR NUMBER(4),
    FISCAL_YEAR NUMBER(4),
    FISCAL_QUARTER NUMBER(1),
    WEEK_OF_YEAR NUMBER(2),
    SEASON VARCHAR2(10)
);
```

Listing 1.5: Query to Set Up Date_Dimension Table

Output:



```
Table DATE_DIMENSION created.
```

Figure 1.7: Output Showing Date_Dimension Table Creation

1.5. Supporting Dimension Tables

1.5.1 Supporting Table for Product_Dimension

A. Product_Category_Dimension

The `Product_Category_Dimension` table supports analysis by product category, enabling detailed queries such as total sales by category or top-performing categories. Each category is uniquely identified and includes optional descriptive details, making it a valuable reference for categorization and product-level analysis within the schema.

Attributes of Product_Category_Dimension:

Attribute	Details
CATEGORY_ID (PK)	A unique identifier for each product category.
CATEGORY_NAME	The name of the category.
CATEGORY_DESCRIPTION	An optional field to provide additional details about the category.

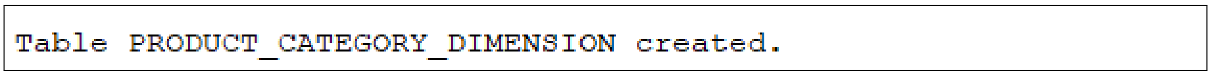
Table 1.6: Description of Attributes in the Product_Category_Dimension Table

SQL Statement to Create the Product_Category_Dimension Table:

```sql
CREATE TABLE Product_Category_Dimension (
    CATEGORY_ID VARCHAR2(4) PRIMARY KEY,
    CATEGORY_NAME VARCHAR2(30),
    CATEGORY_DESCRIPTION VARCHAR2(100)
);
```

Listing 1.6: Query to Set Up Product_Category_Dimension Table

Output:



```
Table PRODUCT_CATEGORY_DIMENSION created.
```

Figure 1.8: Output Showing Product_Category_Dimension Table Creation

B. Brand_Dimension:

The `Brand_Dimension` table enables analysis of total sales by brand and product performance across different brands. Each brand is uniquely identified and can include optional descriptive information, making it a key reference for analyzing brand performance within the schema.

Attributes of Brand_Dimension:

Attribute	Details
BRAND_ID (PK)	A unique identifier for each brand.
BRAND_NAME	The name of the brand associated with the product.
BRAND_DESCRIPTION	An optional field to provide additional details about the brand.

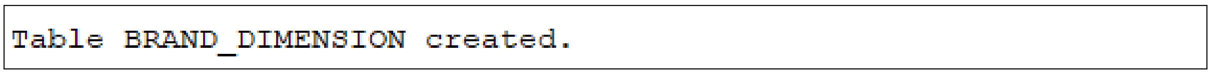
Table 1.7: Description of Attributes in the Brand_Dimension Table

SQL Statement to Create the Brand_Dimension Table:

```sql
CREATE TABLE Brand_Dimension (
    BRAND_ID VARCHAR2(4) PRIMARY KEY,
    BRAND_NAME VARCHAR2(30),
    BRAND_DESCRIPTION VARCHAR2(100)
);
```

Listing 1.7: Query to Set Up Brand_Dimension Table

Output:



```
Table BRAND_DIMENSION created.
```

Figure 1.9: Output Showing Brand_Dimension Table Creation

C. Supplier_Dimension

The `Supplier_Dimension` table enables aggregation and analysis based on supplier details, such as total sales by supplier or performance comparisons among different supplier types. It provides a structured view of each supplier, with a link to additional dimension defining supplier types, allowing for detailed supplier-based analytics. Each supplier is uniquely identified, supporting effective tracking and reporting of supplier-related metrics within the schema.

Attributes of Supplier_Dimension:

Attribute	Details
SUPPLIER_ID (PK)	A unique identifier for each supplier.
SUPPLIER_NAME	The name of the supplier.
SUPPLIER_TYPE_ID (FK)	Links to the type of supplier from the Supplier_Type_Dimension table.

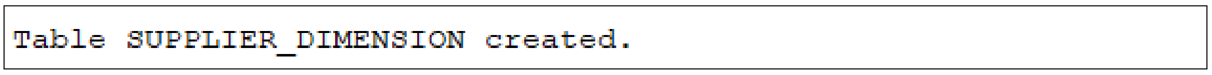
Table 1.8: Description of Attributes in the Supplier_Dimension Table

SQL Statement to Create the Supplier_Dimension Table:

```sql
CREATE TABLE Supplier_Dimension (
    SUPPLIER_ID VARCHAR2(5) PRIMARY KEY,
    SUPPLIER_NAME VARCHAR2(30),
    SUPPLIER_TYPE_ID VARCHAR2(3),
    FOREIGN KEY (SUPPLIER_TYPE_ID) REFERENCES Supplier_Type_Dimension(SUPPLIER_TYPE_ID
        ↪ )
);
```

Listing 1.8: Query to Set Up Supplier_Dimension Table

Output:



```
Table SUPPLIER_DIMENSION created.
```

Figure 1.10: Output Showing Supplier_Dimension Table Creation

1.5.2 Supporting Table for Supplier_Dimension

A. Supplier_Type_Dimension

The `Supplier_Type_Dimension` table provides a detailed classification of supplier types, enabling analysis and segmentation based on supplier categories. It supports queries like total sales by supplier type and comparison of local versus international suppliers. Each supplier type is uniquely identified, making it a critical reference for categorizing suppliers within the `Supplier_Dimension` table.

Attributes of Supplier_Type_Dimension:

Attribute	Details
SUPPLIER_TYPE_ID (PK)	A unique identifier for each supplier type.
SUPPLIER_TYPE_NAME	The name of the supplier type, providing classification of suppliers.

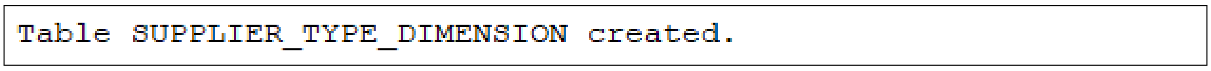
Table 1.9: Description of Attributes in the Supplier_Type_Dimension Table

SQL Statement to Create the Supplier_Type_Dimension Table:

```sql
CREATE TABLE Supplier_Type_Dimension (
    SUPPLIER_TYPE_ID VARCHAR2(3) PRIMARY KEY,
    SUPPLIER_TYPE_NAME VARCHAR2(20)
);
```

Listing 1.9: Query to Set Up Supplier_Type_Dimension Table

Output



```
Table SUPPLIER_TYPE_DIMENSION created.
```

Figure 1.11: Output Showing Supplier_Type_Dimension Table Creation

1.5.3 Supporting Table for Client_Dimension

A. Client_Segment_Dimension

The `Client_Segment_Dimension` table enables analysis based on client segments, facilitating comparisons such as total sales by small businesses versus enterprises. Each segment is uniquely identified, and optional descriptive details can be added to further refine the segment classification, making it a key reference for analyzing client behavior within the schema.

Attributes of Client_Segment_Dimension:

Attribute	Details
CLIENT_SEGMENT_ID (PK)	A unique identifier for each client segment.
SEGMENT_NAME	The name of the segment.
SEGMENT_DESCRIPTION	An optional field to provide additional details about the segment.

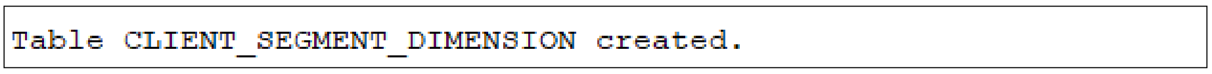
Table 1.10: Description of Attributes in the Client_Segment_Dimension Table

SQL Statement to Create the Client_Segment_Dimension Table:

```sql
CREATE TABLE Client_Segment_Dimension (
    CLIENT_SEGMENT_ID VARCHAR2(3) PRIMARY KEY,
    SEGMENT_NAME VARCHAR2(30),
    SEGMENT_DESCRIPTION VARCHAR2(100)
);
```

Listing 1.10: Query to Set Up Client_Segment_Dimension Table

Output:



```
Table CLIENT_SEGMENT_DIMENSION created.
```

Figure 1.12: Output Showing Client_Segment_Dimension Table Creation

B. Client_Region_Dimension

The `Client_Region_Dimension` table enables aggregations and analysis based on geographic regions, supporting insights such as total sales by region and regional comparisons of business performance. Each region is uniquely identified, with optional descriptive information available to provide further context, making it a critical reference for understanding geographic trends within the schema.

Attributes of Client_Region_Dimension:

Attribute	Details
CLIENT_REGION_ID (PK)	A unique identifier for each geographic region.
REGION_NAME	The name of the region where the client is located.
REGION_DESCRIPTION	An optional field for additional details about the region.

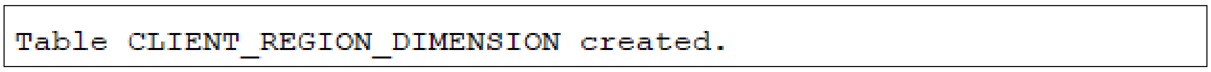
Table 1.11: Description of Attributes in the Client_Region_Dimension Table

SQL Statement to Create the Client_Region_Dimension Table:

```sql
CREATE TABLE Client_Region_Dimension (
    CLIENT_REGION_ID VARCHAR2(3) PRIMARY KEY,
    REGION_NAME VARCHAR2(30),
    REGION_DESCRIPTION VARCHAR2(100)
);
```

Listing 1.11: Query to Set Up Client_Region_Dimension Table

Output:



```
Table CLIENT_REGION_DIMENSION created.
```

Figure 1.13: Output Showing Client_Region_Dimension Table Creation

1.5.4 Supporting Table for Channel_Dimension

A. Channel_Type_Dimension

The `Channel_Type_Dimension` table enables analysis based on the type of sales channel, supporting comparisons such as online versus in-store sales. Each channel type is uniquely identified, and optional descriptive details can be included to provide further context, making it a valuable reference for categorizing sales channels within the schema..

Attributes of Channel_Type_Dimension:

Attribute	**Details**
CHANNEL_TYPE_ID (PK)	A unique identifier for each channel type.
CHANNEL_TYPE_NAME	The name of the channel type.
CHANNEL_TYPE_DESCRIPTION	An optional field for additional details about the channel type.

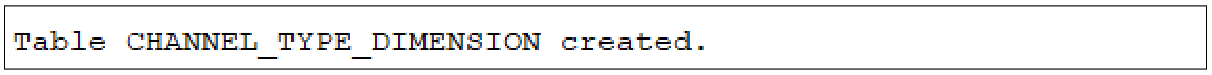
Table 1.12: Description of Attributes in the Channel_Type_Dimension Table

SQL Statement to Create the Channel_Type_Dimension Table:

```sql
CREATE TABLE Channel_Type_Dimension (
    CHANNEL_TYPE_ID VARCHAR2(3) PRIMARY KEY,
    CHANNEL_TYPE_NAME VARCHAR2(20),
    CHANNEL_TYPE_DESCRIPTION VARCHAR2(100)
);
```

Listing 1.12: Query to Set Up Channel_Type_Dimension Table

Output:



```
Table CHANNEL_TYPE_DIMENSION created.
```

Figure 1.14: Output Showing Channel_Type_Dimension Table Creation

B. Channel_Region_Dimension

The `Channel_Region_Dimension` table supports analysis of sales by the geographic region associated with each sales channel, enabling comparisons of sales performance across different areas. Each region is uniquely identified, and optional descriptive information can be added to further contextualize the channel's region, making it a key reference for understanding regional sales trends within the schema.

Attributes of Channel_Region_Dimension:

Attribute	Details
CHANNEL_REGION_ID (PK)	A unique identifier for each channel region.
REGION_NAME	The name of the region.
REGION_DESCRIPTION	An optional field for providing additional details about the region.

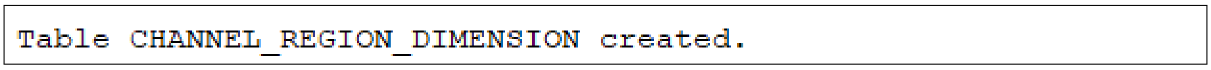
Table 1.13: Description of Attributes in the Channel_Region_Dimension Table

SQL Statement to Create the Channel_Region_Dimension Table:

```sql
CREATE TABLE Channel_Region_Dimension (
    CHANNEL_REGION_ID VARCHAR2(3) PRIMARY KEY,
    CHANNEL_REGION_NAME VARCHAR2(30),
    REGION_DESCRIPTION VARCHAR2(100)
);
```

Listing 1.13: Query to Set Up Channel_Region_Dimension Table

Output:



```
Table CHANNEL_REGION_DIMENSION created.
```

Figure 1.15: Output Showing Channel_Region_Dimension Table Creation

1.6. Aggregation Possibilities

By normalizing the dimensions and adopting a snowflake schema, the data warehouse is optimized to support a diverse array of aggregation possibilities. To illustrate these capabilities for the sake of this report, tables are populated with sample data. However, the script for populating the fact and dimension tables with sample data is not submitted with the assignment and not included in this report. In a real business scenario, when the tables are populated with actual business data, analytical queries such as those below can be executed to generate the required business insights.

1.6.1 Total Sales by Product Category

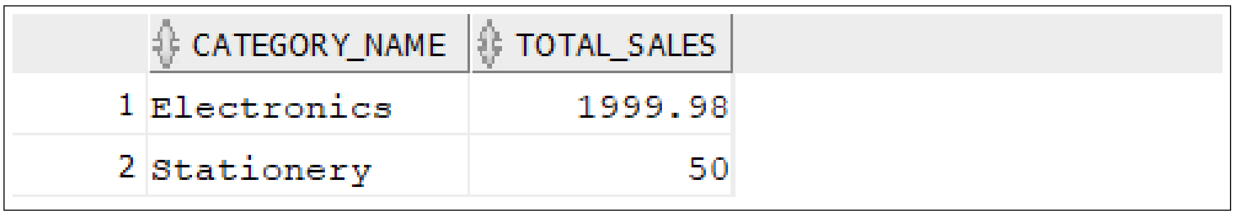
The following query calculates the total sales for each product category by summing the `TOTAL_SALES_VALUE` field from the `Sales_Fact` table. It joins the `Sales_Fact` table with the `Product_Dimension` and `Product_Category_Dimension` tables to access the relevant category information. The results are then grouped by each product category to provide a complete view of sales performance.

SQL Statement:

```sql
SELECT
    pcd.CATEGORY_NAME,
    SUM(sf.TOTAL_SALES_VALUE) AS TOTAL_SALES
FROM
    Sales_Fact sf
    JOIN Product_Dimension pd ON sf.PRODUCT_ID = pd.PRODUCT_ID
    JOIN Product_Category_Dimension pcd ON pd.CATEGORY_ID = pcd.CATEGORY_ID
GROUP BY
    pcd.CATEGORY_NAME;
```

Listing 1.14: Query to Calculate Total Sales by Product Category

Output:



	CATEGORY_NAME	TOTAL_SALES
1	Electronics	1999.98
2	Stationery	50

Figure 1.16: Output Showing Total Sales by Product Category

1.6.2 Total Sales by Client Segment

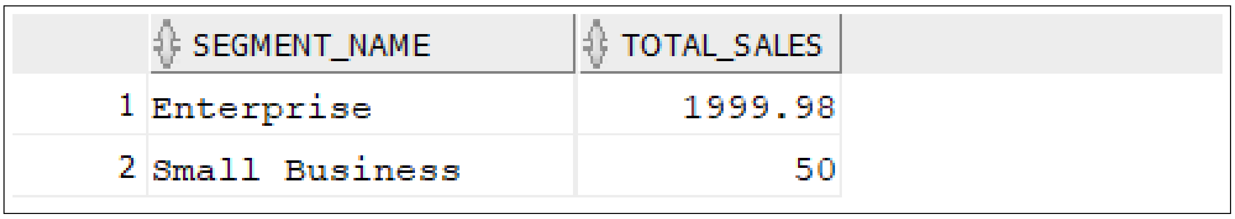
The following query calculates the total sales for each client segment by summing the `TOTAL_SALES_VALUE` field from the `Sales_Fact` table. It joins the `Sales_Fact` table with the `Client_Dimension` and `Client_Segment_Dimension` tables to access the relevant client segment details. The results are then grouped by each client segment, providing insight into sales distribution across different customer categories.

SQL Statement:

```sql
SELECT
    csd.SEGMENT_NAME,
    SUM(sf.TOTAL_SALES_VALUE) AS TOTAL_SALES
FROM
    Sales_Fact sf
    JOIN Client_Dimension cd ON sf.CLIENT_ID = cd.CLIENT_ID
    JOIN Client_Segment_Dimension csd ON cd.CLIENT_SEGMENT_ID = csd.CLIENT_SEGMENT_ID
GROUP BY
    csd.SEGMENT_NAME;
```

Listing 1.15: Query to Calculate Total Sales by Client Segment

Output:



	SEGMENT_NAME	TOTAL_SALES
1	Enterprise	1999.98
2	Small Business	50

Figure 1.17: Output Showing Total Sales by Client Segment

1.6.3 Total Sales by Channel Type

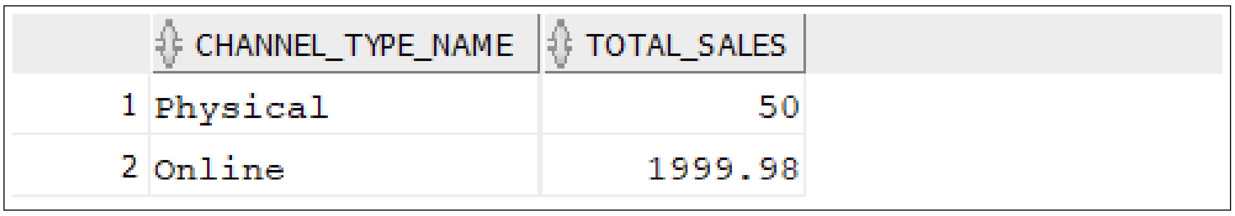
The following query calculates the total sales for each channel type by summing the `TOTAL_SALES_VALUE` field from the `Sales_Fact` table. It joins the `Sales_Fact` table with the `Channel_Dimension` and `Channel_Type_Dimension` tables to access the relevant channel type information. The results are then grouped by each channel type, providing insight into sales performance across different sales channels.

SQL Statement:

```sql
SELECT
    ctd.CHANNEL_TYPE_NAME,
    SUM(sf.TOTAL_SALES_VALUE) AS TOTAL_SALES
FROM
    Sales_Fact sf
    JOIN Channel_Dimension cd ON sf.CHANNEL_ID = cd.CHANNEL_ID
    JOIN Channel_Type_Dimension ctd ON cd.CHANNEL_TYPE_ID = ctd.CHANNEL_TYPE_ID
GROUP BY
    ctd.CHANNEL_TYPE_NAME;
```

Listing 1.16: Query to Calculate Total Sales by Channel Type

Output:



	CHANNEL_TYPE_NAME	TOTAL_SALES
1	Physical	50
2	Online	1999.98

Figure 1.18: Output Showing Total Sales by Channel Type

1.6.4 Seasonal Trends in Sales

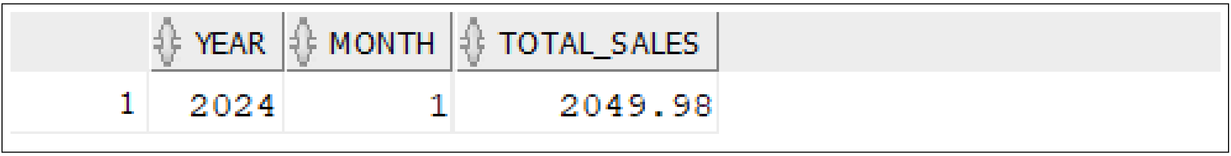
The following query analyzes seasonal trends in sales by calculating the total sales for each year and month. It joins the `Sales_Fact` table with the `Date_Dimension` table to access the date details and then uses the `EXTRACT` function to group the results by year and month. This allows for the identification of patterns and fluctuations in sales over time, providing valuable insights into seasonal performance.

SQL Statement:

```sql
SELECT
    EXTRACT(YEAR FROM dd.DATE_COLUMN) AS YEAR,
    EXTRACT(MONTH FROM dd.DATE_COLUMN) AS MONTH,
    SUM(sf.TOTAL_SALES_VALUE) AS TOTAL_SALES
FROM
    Sales_Fact sf
    JOIN Date_Dimension dd ON sf.DATE_KEY = dd.DATE_KEY
GROUP BY
    EXTRACT(YEAR FROM dd.DATE_COLUMN),
    EXTRACT(MONTH FROM dd.DATE_COLUMN)
ORDER BY
    YEAR, MONTH;
```

Listing 1.17: Query to Calculate Seasonal Trends in Sales

Output:



	YEAR	MONTH	TOTAL_SALES
1	2024	1	2049.98

Figure 1.19: Output Showing Seasonal Trends in Sales

1.6.5 Total Sales by Product Brand and Sales Channel

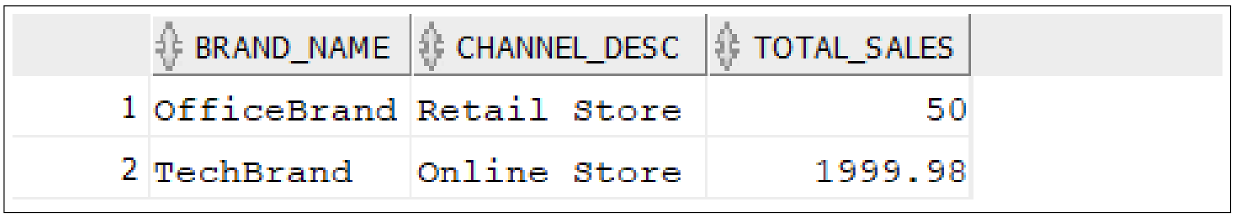
The following query calculates the total sales by combining product brand and sales channel information. It joins the `Sales_Fact` table with the `Product_Dimension`, `Brand_Dimension`, and `Channel_Dimension` tables to retrieve the relevant brand and channel details. The results are grouped by both the brand name and sales channel description, providing insights into how different brands perform across various sales channels.

SQL Statement:

```sql
SELECT
    bd.BRAND_NAME,
    cd.CHANNEL_DESC,
    SUM(sf.TOTAL_SALES_VALUE) AS TOTAL_SALES
FROM
    Sales_Fact sf
    JOIN Product_Dimension pd ON sf.PRODUCT_ID = pd.PRODUCT_ID
    JOIN Brand_Dimension bd ON pd.BRAND_ID = bd.BRAND_ID
    JOIN Channel_Dimension cd ON sf.CHANNEL_ID = cd.CHANNEL_ID
GROUP BY
    bd.BRAND_NAME,
    cd.CHANNEL_DESC;
```

Listing 1.18: Query to Calculate Total Sales by Product Brand and Sales Channel

Output:



	BRAND_NAME	CHANNEL_DESC	TOTAL_SALES
1	OfficeBrand	Retail Store	50
2	TechBrand	Online Store	1999.98

Figure 1.20: Output Showing Total Sales by Product Brand and Sales Channel

1.6.6 Summary

Beyond these specific analyses, this snowflake schema supports a broad range of additional aggregations. For example, OfficeProducts can analyze total sales by specific product suppliers, compare sales performance across different regions, evaluate trends in sales across various fiscal years, or assess the impact of promotions on sales performance, and much more. This flexibility ensures that the data warehouse remains a valuable tool for ongoing business intelligence and strategic planning.

1.7. Conclusion

In conclusion, it can be said that this design establishes a strong foundation for OfficeProducts to optimize its operations, from marketing and promotions to logistics and inventory management. Normalizing the dimension tables and linking them to a central fact table enables powerful cross-dimensional aggregations, providing deep insights into product performance, client behavior, and sales trends. The schema not only minimizes redundancy but also improves query performance when aggregating large volumes of sales data, facilitating strategic decision-making with greater confidence.

Section B

2.1. Part A - Short-answer questions

2.1.1 Question 1

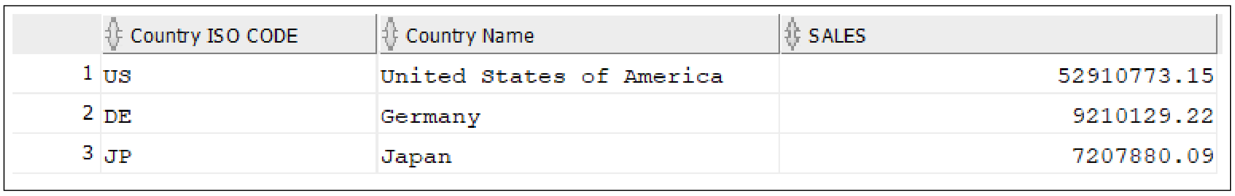
The following query identifies the top three countries with the highest total sales. It joins the `Sales` table with the `Customers` and `Countries` tables. The query then groups the results by `country_iso_code` and `country_name` to calculate the total sales (`amount_sold`) for each country. After grouping, the results are sorted in descending order based on the total sales amount, ensuring that countries with the highest sales are displayed first. Finally, the query uses the `FETCH FIRST 3 ROWS ONLY` clause to limit the output to only the top three countries.

SQL Statement:

```sql
SELECT
    c.country_iso_code AS "Country ISO CODE",
    c.country_name AS "Country Name",
    SUM(s.amount_sold) AS "SALES"
FROM
    sh.sales s
JOIN
    sh.customers cu ON s.cust_id = cu.cust_id
JOIN
    sh.countries c ON cu.country_id = c.country_id
GROUP BY
    c.country_iso_code, c.country_name
ORDER BY
    "SALES" DESC
FETCH FIRST 3 ROWS ONLY;
```

Listing 2.1: Query to Show Top 3 Countiries With the Highest Total Sales

Output:



```
     Country ISO CODE   Country Name                 SALES
1    US                 United States of America     52910773.15
2    DE                 Germany                       9210129.22
3    JP                 Japan                         7207880.09
```

Figure 2.1: Output Showing Top 3 Countiries With the Highest Total Sales

The results show that the United States has the highest total sales at approximately 52.91 million USD, far surpassing Germany and Japan, which have 9.21 million USD and 7.21 million USD, respectively. This indicates that the US market is the primary revenue driver, likely due to its larger consumer base and strong brand presence. In comparison, Germany and Japan, while still significant contributors, generate much lower sales, suggesting opportunities for growth and increased market penetration in these regions.

2.1.2 Question 2

The following query identifies the most sold products in terms of quantity in the United States for each year from 1998 to 2001. It combines data from the `Sales`, `Products`, `Customers`, `Countries`, and `Times` tables. The query first filters the results to include only those sales that occurred in the United States (`country_iso_code = 'US'`) and within the specified time frame of 1998 to 2001. The results are then grouped by `calendar_year` and `prod_name` to calculate the total quantity sold for each product in each year.

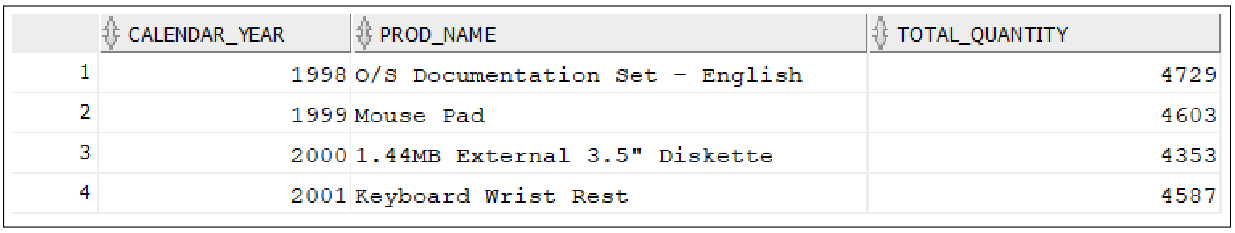
To identify the most sold product in each year, the `HAVING` clause is used along with a subquery that compares each product's total quantity against the maximum quantity sold for that year. This ensures that only the top-selling product for each year is included in the final result set. The query outputs the `prod_name`, `calendar_year`, and `total_quantity` sold, sorted first by year and then by quantity in descending order.

SQL Statement:

```
SELECT
    t.calendar_year AS CALENDAR_YEAR,
    p.prod_name AS PROD_NAME,
    SUM(s.quantity_sold) AS TOTAL_QUANTITY
FROM
    sh.sales s
JOIN
    sh.products p ON s.prod_id = p.prod_id
JOIN
    sh.customers cust ON s.cust_id = cust.cust_id
JOIN
    sh.countries c ON cust.country_id = c.country_id
JOIN
    sh.times t ON s.time_id = t.time_id
WHERE
    c.country_iso_code = 'US'
    AND t.calendar_year BETWEEN 1998 AND 2001
GROUP BY
    t.calendar_year, p.prod_name
HAVING
    SUM(s.quantity_sold) = (
        SELECT MAX(SUM(s2.quantity_sold))
        FROM sh.sales s2
        JOIN sh.products p2 ON s2.prod_id = p2.prod_id
        JOIN sh.customers cust2 ON s2.cust_id = cust2.cust_id
        JOIN sh.countries c2 ON cust2.country_id = c2.country_id
        JOIN sh.times t2 ON s2.time_id = t2.time_id
        WHERE
            c2.country_iso_code = 'US'
            AND t2.calendar_year = t.calendar_year
        GROUP BY
            p2.prod_name
    )
ORDER BY
    t.calendar_year, TOTAL_QUANTITY DESC;
```

Listing 2.2: Query to Find the Top Product by Quantity Sold for Each Year from 1998 through 2001 in the US

Output:



	CALENDAR_YEAR	PROD_NAME	TOTAL_QUANTITY
1	1998	O/S Documentation Set - English	4729
2	1999	Mouse Pad	4603
3	2000	1.44MB External 3.5" Diskette	4353
4	2001	Keyboard Wrist Rest	4587

Figure 2.2: Output Showing the Top Product by Quantity Sold for Each Year from 1998 through 2001 in the US.

From 1998 to 2001, the most sold products in the US varied each year, highlighting shifts in consumer demand and preferences. In 1998, the O/S Documentation Set - English was the top-selling item with 4,729 units sold. The following year, in 1999, the Mouse Pad took the lead with 4,603 units sold. By 2000, demand shifted to storage solutions, with the 1.44MB External 3.5" Diskette being the most sold item at 4,353 units. In 2001, ergonomic accessories gained popularity as the Keyboard Wrist Rest topped sales with 4,587 units. This trend reflects the evolving nature of technology and workplace needs during that time.

2.1.3 Question 3

The following query analyzes the transaction details for the top-selling product of the year 2001. It starts by defining a Common Table Expression (CTE) called `Max_Product_2001`, which calculates the total sales amount for each product sold in 2001 by joining the `sales` and `times` tables. The query filters sales records to include only the year 2001 and groups them by `prod_id`. It then orders the products by total sales in descending order and uses the `FETCH FIRST 1 ROWS ONLY` clause to select the product with the highest sales, thereby identifying the top-selling product of the year.

The second CTE, `Product_Transaction_Details`, focuses on gathering transaction details specifically for the top-selling product identified earlier. This CTE joins the `sales` and `channels` tables, incorporating additional details about the sales channels. It groups the data by `prod_id`, `channel_id`, and `channel_desc`, then computes the total number of transactions (`num_trans`) and the total quantity sold (`total_quantity`) for each channel.

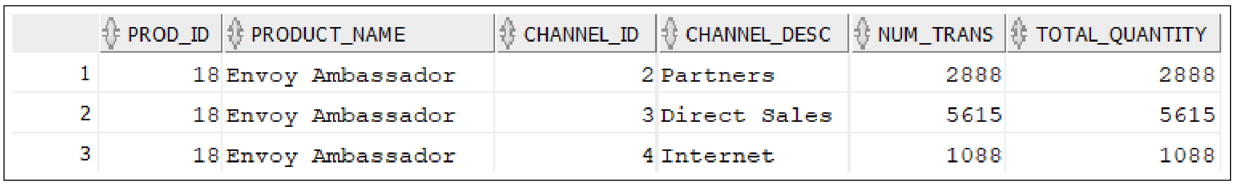
Finally, the main query selects and displays the transaction details of the top-selling product along with its associated sales channels. The result set includes the `product_id`, `product_name`, `channel_id`, `channel_description`, number of transactions, and total quantity sold, providing a panoramic view of the top product's performance across various sales channels.

SQL Statement:

```sql
WITH Max_Product_2001 AS (
    SELECT
        s.prod_id,
        SUM(s.amount_sold) AS total_sales
    FROM
        sh.sales s
    JOIN
        sh.times t ON s.time_id = t.time_id
    WHERE
        EXTRACT(YEAR FROM t.time_id) = 2001
    GROUP BY
        s.prod_id
    ORDER BY
        total_sales DESC
    FETCH FIRST 1 ROWS ONLY
),
Product_Transaction_Details AS (
    SELECT
        s.prod_id,
        s.channel_id,
        c.channel_desc,
        COUNT(s.prod_id) AS num_trans,
        SUM(s.quantity_sold) AS total_quantity
    FROM
        sh.sales s
    JOIN
        sh.channels c ON s.channel_id = c.channel_id
    JOIN
        Max_Product_2001 mp ON s.prod_id = mp.prod_id
    GROUP BY
        s.prod_id, s.channel_id, c.channel_desc
)
SELECT
    p.prod_id AS "PROD_ID",
    p.prod_name AS "PRODUCT_NAME",
    pt.channel_id AS "CHANNEL_ID",
    pt.channel_desc AS "CHANNEL_DESC",
    pt.num_trans AS "NUM_TRANS",
    pt.total_quantity AS "TOTAL_QUANTITY"
FROM
    Product_Transaction_Details pt
JOIN
    sh.products p ON pt.prod_id = p.prod_id;
```

Listing 2.3: Query to Retrieve Transaction Details for the Top-Selling Product of 2001

Output:



	PROD_ID	PRODUCT_NAME	CHANNEL_ID	CHANNEL_DESC	NUM_TRANS	TOTAL_QUANTITY
1	18	Envoy Ambassador	2	Partners	2888	2888
2	18	Envoy Ambassador	3	Direct Sales	5615	5615
3	18	Envoy Ambassador	4	Internet	1088	1088

Figure 2.3: Output Showing Transaction Details for the Top-Selling Product of 2001

The product with the highest sales revenue in 2001 was the Envoy Ambassador. The item was sold through three channels: Partners (Channel ID: 2) with 2,888 transactions, Direct Sales (Channel ID: 3) with 5,615 transactions, and Internet (Channel ID: 4) with 1,088 transactions. The majority of the sales came through the Direct Sales channel, indicating a stronger performance and preference for direct customer engagement over other channels.

2.1.4 Question 4

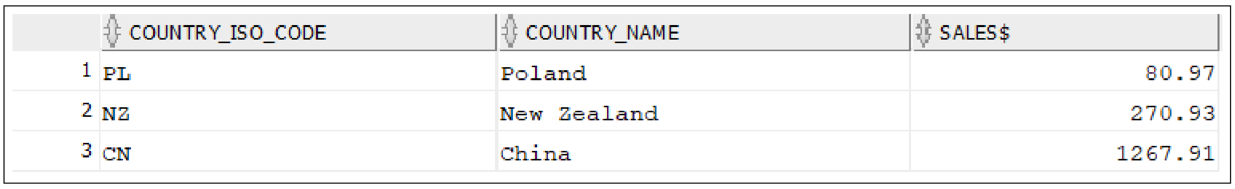
The following query identifies the three countries with the lowest total sales in the year 1998. It first combines data from the `Sales`, `Customers`, `Countries`, and `Times` tables. The statement then filters the results to include only sales that occurred in the year 1998 using the condition `EXTRACT(YEAR FROM t.time_id)`. Afterward, the data is grouped by `country_iso_code` and `country_name` to calculate the total sales for each country. The results are sorted in ascending order based on the total sales amount (`SALES$`), ensuring that countries with the lowest sales are listed first. Finally, the `FETCH FIRST 3 ROWS ONLY` clause is used to limit the output to the top three countries with the lowest total sales in 1998.

SQL Statement:

```sql
SELECT
    c.country_iso_code AS "COUNTRY_ISO_CODE",
    c.country_name AS "COUNTRY_NAME",
    SUM(s.amount_sold) AS "SALES$"
FROM
    sh.sales s
JOIN
    sh.customers cu ON s.cust_id = cu.cust_id
JOIN
    sh.countries c ON cu.country_id = c.country_id
JOIN
    sh.times t ON s.time_id = t.time_id
WHERE
    EXTRACT(YEAR FROM t.time_id) = 1998
GROUP BY
    c.country_iso_code, c.country_name
ORDER BY
    "SALES$" ASC
FETCH FIRST 3 ROWS ONLY;
```

Listing 2.4: Query to Retrieve the Bottom 3 Countries by Sales for the Year 1998

Output:



	COUNTRY_ISO_CODE	COUNTRY_NAME	SALES$
1	PL	Poland	80.97
2	NZ	New Zealand	270.93
3	CN	China	1267.91

Figure 2.4: Output Showing the Bottom 3 Countries by Sales for the Year 1998

The three countries with the poorest sales performance in 1998 were Poland, New Zealand, and China. Poland had the lowest sales at 80.97 USD, followed by New Zealand with 270.93 USD, and China with 1,267.91 USD. This suggests that these markets showed minimal engagement or purchasing activity during the year.

2.1.5 Question 5

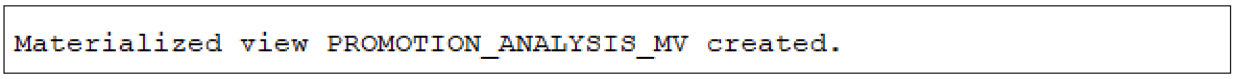
The following query creates a materialized view named `Promotion_Analysis_mv` to provide a sales analysis by product for each promotion. It joins the `Sales` table with the `Products` and `Promotions` tables to associate each sale with the relevant product and promotion details. The results are grouped by `promo_id` and `prod_id` to calculate the total sales amount for each product under every promotion. This materialized view is built immediately and set to refresh completely when updated, ensuring the data is always up-to-date.

SQL Statement:

```sql
CREATE MATERIALIZED VIEW Promotion_Analysis_mv
BUILD IMMEDIATE
REFRESH COMPLETE
AS
SELECT
    s.promo_id,
    s.prod_id,
    SUM(s.amount_sold) AS total_sales
FROM
    sh.sales s
JOIN
    sh.products p ON s.prod_id = p.prod_id
JOIN
    sh.promotions pr ON s.promo_id = pr.promo_id
GROUP BY
    s.promo_id,
    s.prod_id;
```

Listing 2.5: Query to Create the Materialized View

Output:



```
Materialized view PROMOTION_ANALYSIS_MV created.
```

Figure 2.5: Output Showing Materialized View Created

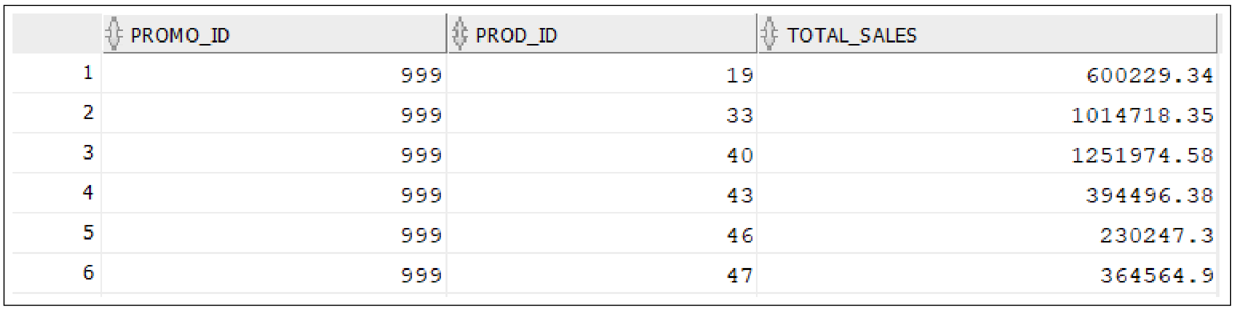
In addition, a subsequent query selects the columns `promo_id`, `prod_id`, and `total_sales` from the `Promotion_Analysis_mv` table, displaying each promotion's ID, the associated product ID, and the corresponding total sales.

SQL Statement:

```sql
SELECT
    promo_id,
    prod_id,
    total_sales
FROM
    Promotion_Analysis_mv
```

Listing 2.6: Query to Retrieve Total Sales by Promotion and Product ID

Output:



	PROMO_ID	PROD_ID	TOTAL_SALES
1	999	19	600229.34
2	999	33	1014718.35
3	999	40	1251974.58
4	999	43	394496.38
5	999	46	230247.3
6	999	47	364564.9

Figure 2.6: Output Showing Total Sales by Promotion and Product ID

2.2. Part B. Long-answer questions

2.2.1 Question 6

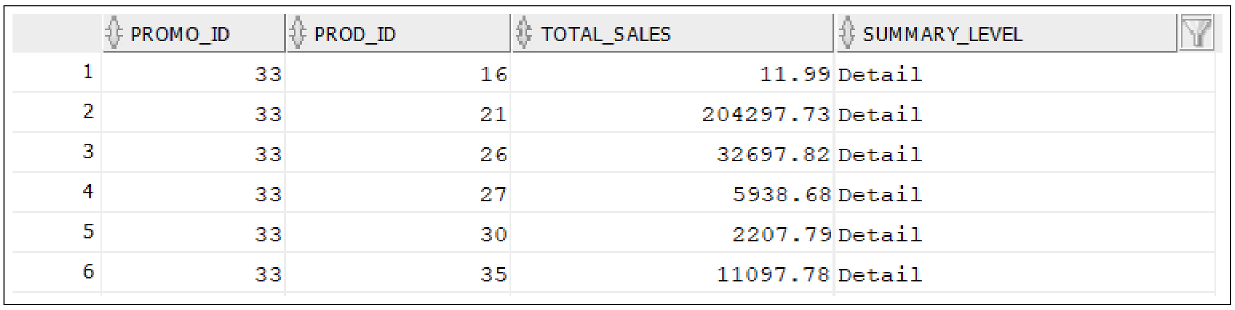
The following query utilizes the `Promotion_Analysis_mv` materialized view to analyze total sales at various aggregation levels using the `ROLLUP` function. It calculates sales totals for each promotion and product combination, as well as subtotals for each product, and a grand total for all data. The `CASE` statement categorizes the results into `"Detail"`, `"Product Total"`, or `"Grand Total"` for easier interpretation. This approach helps management understand sales distribution across promotions and products, providing valuable insights into performance trends.

SQL Statement:

```
SELECT
    promo_id,
    prod_id,
    SUM(total_sales) AS total_sales,
    CASE
        WHEN GROUPING(promo_id) = 1 AND GROUPING(prod_id) = 1 THEN 'Grand Total'
        WHEN GROUPING(prod_id) = 1 THEN 'Product Total'
        ELSE 'Detail'
    END AS summary_level
FROM
    Promotion_Analysis_mv
GROUP BY
    ROLLUP(promo_id, prod_id)
ORDER BY
    promo_id, prod_id;
```

Listing 2.7: Query to Calculate Total Sales by Promotion and Product with Rollup

Output:



	PROMO_ID	PROD_ID	TOTAL_SALES	SUMMARY_LEVEL
1	33	16	11.99	Detail
2	33	21	204297.73	Detail
3	33	26	32697.82	Detail
4	33	27	5938.68	Detail
5	33	30	2207.79	Detail
6	33	35	11097.78	Detail

Figure 2.7: Output Showing Sales Totals by Promotion and Product with Rollup

The query result provides a detailed breakdown of sales by promotion and product, followed by subtotals for each promotion and an overall grand total.The sales details for each `promo_id` are as follows:

- Promotion 33: Individual product sales sum up to a Product Total of $277,426.26.
- Promotion 350: The product sales under this promotion amount to a Product Total of $2,199,380.90.
- Promotion 351: This promotion achieves a Product Total of $1,224,503.26.
- Promotion 999: This is the highest contributor, with total sales for all its products reaching a Product Total of $94,504,520.79.

These subtotals are then combined to provide the Grand Total sales across all promotions, which is $98,205,831.21. This summary helps in understanding the overall contribution of each promotion to the company's sales and identifying which promotions were the most successful.

2.2.2 Question 7

This following query identifies the top five product subcategories by total sales using data from the materialized view `sh.fweek_pscat_sales_mv` within the `SH` schema. The `sh.fweek_pscat_sales_mv` table aggregates sales data at the subcategory level, making it a suitable choice for this analysis because it provides pre-calculated sales metrics, improving query performance and efficiency.

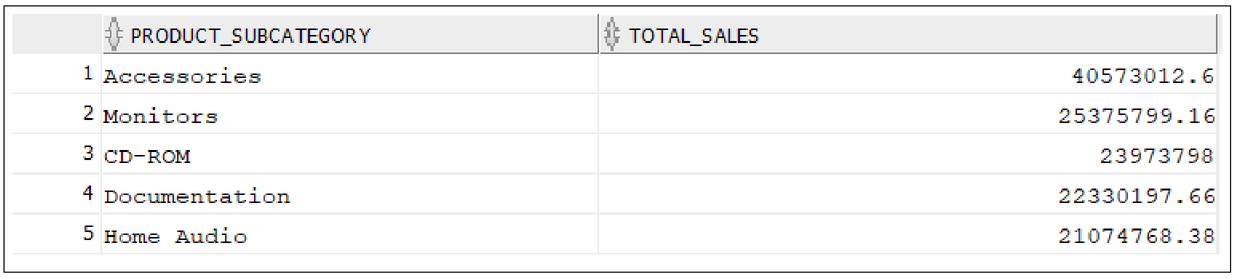
The query joins `sh.fweek_pscat_sales_mv` with the `sh.products` table to ensure that the product subcategory names are retrieved and matched correctly. By summing the sales values (`SUM(fps.dollars)`) and grouping by `prod_subcategory`, the query calculates the total sales for each subcategory and then ranks them in descending order. The `FETCH FIRST 5 ROWS ONLY` clause is used to limit the result set to the top five subcategories, providing management with a clear view of which subcategories are contributing the most to overall sales. This insight can help in strategic decision-making, such as optimizing inventory or marketing focus.

SQL Statement:

```sql
SELECT
    p.prod_subcategory AS product_subcategory,
    SUM(fps.dollars) AS total_sales
FROM
    sh.fweek_pscat_sales_mv fps
JOIN
    sh.products p ON fps.prod_subcategory = p.prod_subcategory
GROUP BY
    p.prod_subcategory
ORDER BY
    total_sales DESC
FETCH FIRST 5 ROWS ONLY;
```

Listing 2.8: Query to Retrieve the Top 5 Product Subcategories by Total Sales

Output:



	PRODUCT_SUBCATEGORY	TOTAL_SALES
1	Accessories	40573012.6
2	Monitors	25375799.16
3	CD-ROM	23973798
4	Documentation	22330197.66
5	Home Audio	21074768.38

Figure 2.8: Output Showing the Top 5 Product Subcategories by Total Sales

The analysis shows that Accessories lead in total sales, generating over $40.57 million, making it the highest-grossing product subcategory. The robust sales figures suggest that

Accessories are a popular choice among customers, possibly due to a combination of factors such as product variety, price points, or compatibility with other products in the catalog. Monitors and CD-ROM follow, contributing $25.38 million and $23.97 million, respectively, indicating a strong demand in these categories as well. This could be attributed to trends in remote work, gaming, or multimedia content consumption. Documentation and Home Audio complete the top five, with sales figures of $22.33 million and $21.07 million. The strong sales in Documentation indicate that there is still considerable value in providing high-quality manuals, guides, or reference materials. On the other hand, Home Audio's performance points to a solid market for audio equipment, potentially driven by trends in home entertainment and smart speaker adoption. These findings provide valuable insight for management to prioritize high-performing subcategories, optimize inventory management, and identify potential areas for strategic investment or marketing efforts.