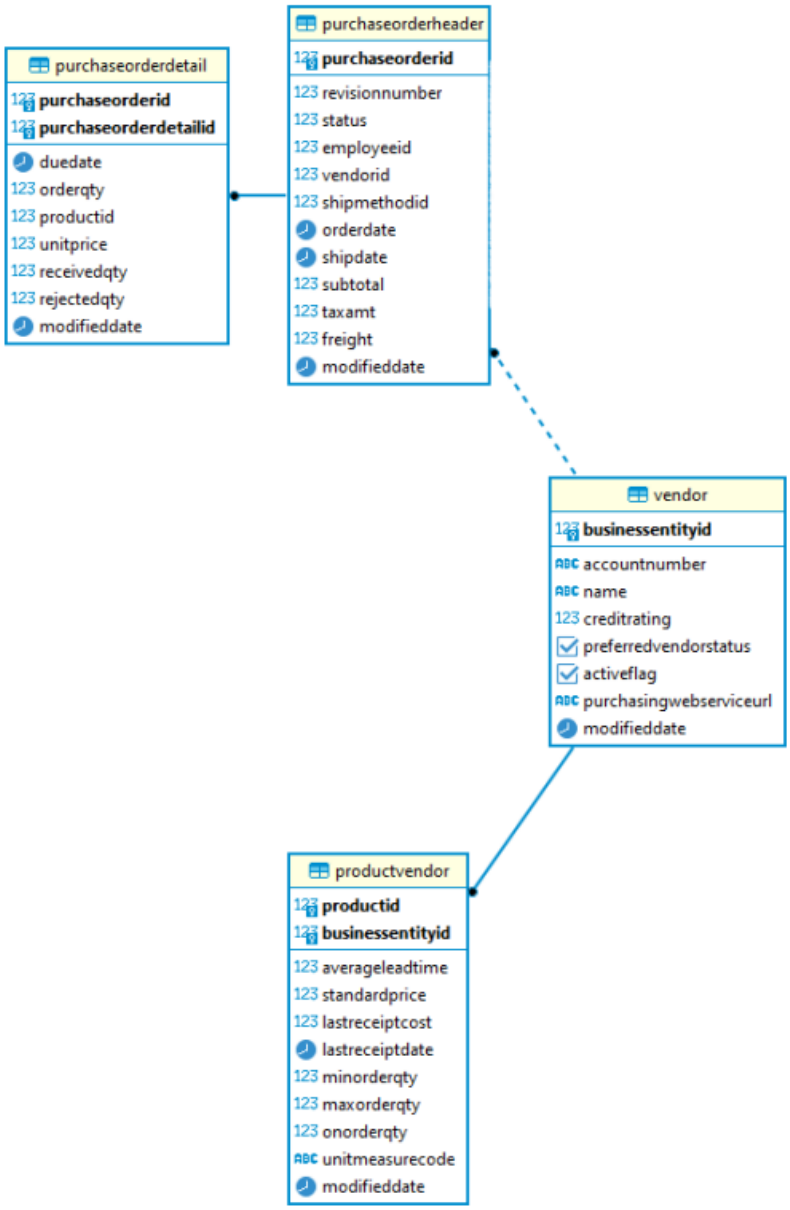
Raphaël Veron
Hugo Verley

Introduction

This report presents the development and optimization of a relational database within Oracle SQL Developer, along with the subsequent visualization using Power BI dashboards. It aims to provide insights into advanced database management and effective data presentation strategies.

Database Schema Design

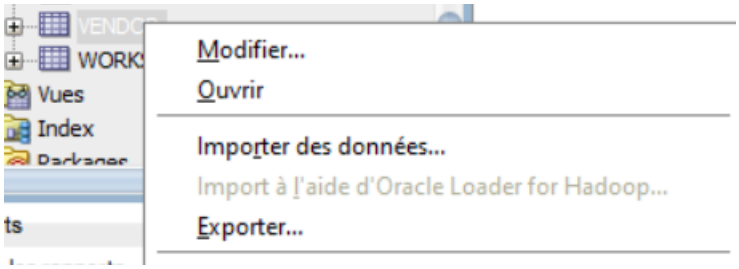
A detailed schema was established, comprising tables such as **Vendor**, **ProductVendor**, **PurchaseOrderDetail**, and **PurchaseOrderHeader**. Relations and constraints were implemented to maintain data integrity.



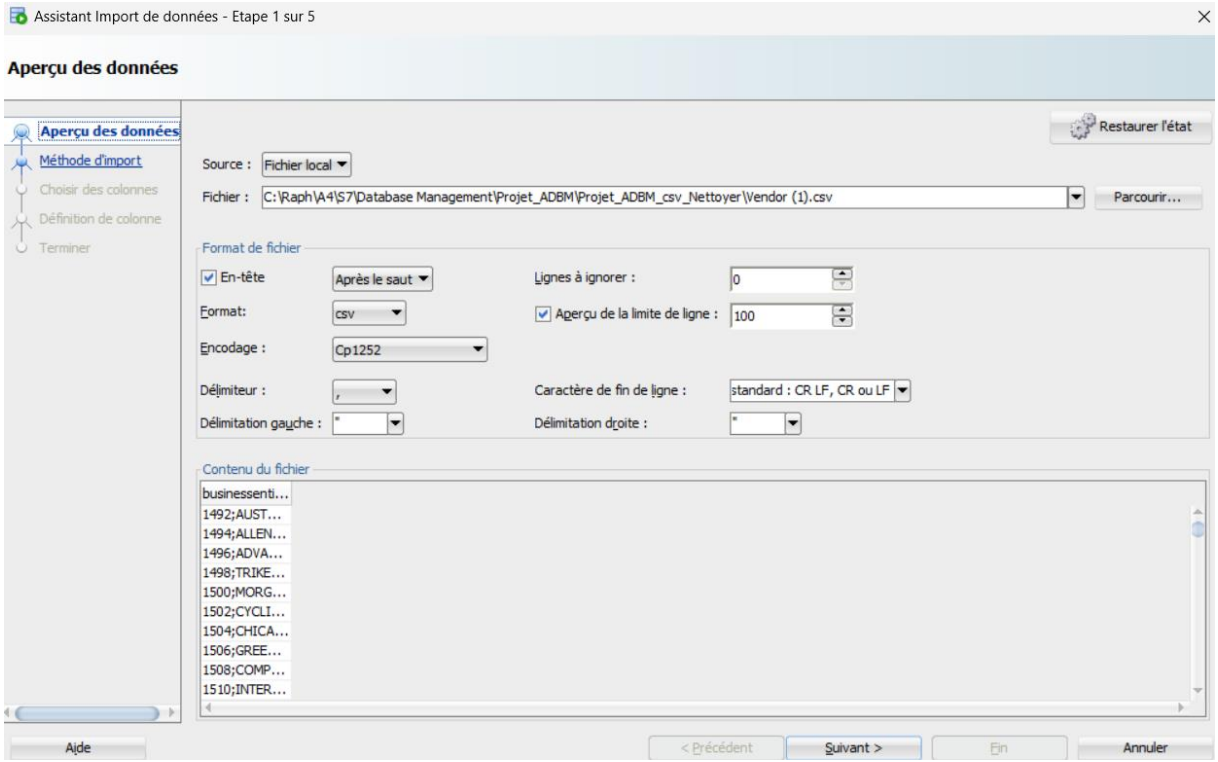
```
--TABLE Vendor
CREATE TABLE Vendor (
    BusinessEntityID NUMBER PRIMARY KEY,
    AccountNumber VARCHAR2(50),
    Name VARCHAR2(100),
    CreditRating NUMBER,
    PreferredVendorStatus VARCHAR2(10),
    ActiveFlag VARCHAR2(10),
    PurchasingWebServiceURL VARCHAR2(150),
    ModifiedDate DATE,
    FOREIGN KEY (BusinessEntityID) REFERENCES Vendor(BusinessEntityID)
);

--Table ProductVendor
CREATE TABLE ProductVendor (
    ProductID NUMBER,
    BusinessEntityID NUMBER,
    AverageLeadTime NUMBER,
    StandardPrice NUMBER,
    LastReceiptCost NUMBER,
    LastReceiptDate DATE,
    MinOrderQty NUMBER,
    MaxOrderQty NUMBER,
    OnOrderQty NUMBER,
    UnitMeasureCode VARCHAR2(10),
    ModifiedDate DATE,
    PRIMARY KEY (ProductID, BusinessEntityID),
    FOREIGN KEY (BusinessEntityID) REFERENCES Vendor(BusinessEntityID)
);

--Table PurchaseOrderDetail
CREATE TABLE PurchaseOrderDetail (
    PurchaseOrderID NUMBER,
    PurchaseOrderDetailID NUMBER PRIMARY KEY,
    DueDate DATE,
    OrderQty NUMBER,
    ProductID NUMBER,
    BusinessEntityID NUMBER,
    UnitPrice NUMBER,
    ReceivedQty NUMBER,
    RejectedQty NUMBER,
    ModifiedDate DATE,
    FOREIGN KEY (ProductID, BusinessEntityID) REFERENCES ProductVendor(Produ
);
```

```
--Table PurchaseOrderHeader
CREATE TABLE PurchaseOrderHeader (
    PurchaseOrderID NUMBER PRIMARY KEY,
    RevisionNumber NUMBER,
    Status NUMBER,
    EmployeeID NUMBER,
    VendorID NUMBER,
    ShipMethodID NUMBER,
    OrderDate DATE,
    ShipDate DATE,
    Subtotal NUMBER,
    TaxAmt NUMBER,
    Freight NUMBER,
    ModifiedDate DATE,
    FOREIGN KEY (VendorID) REFERENCES Vendor(BusinessEntityID)
);
```

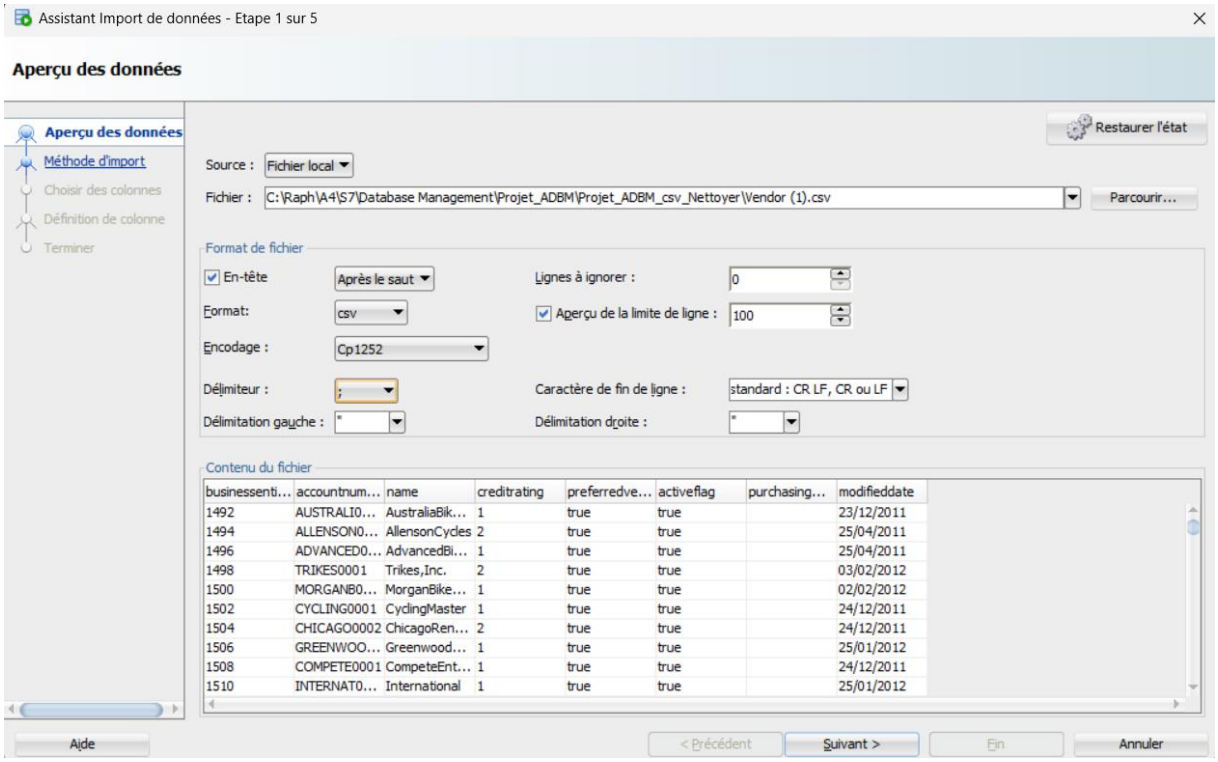
To data to our tables we follow the process below :



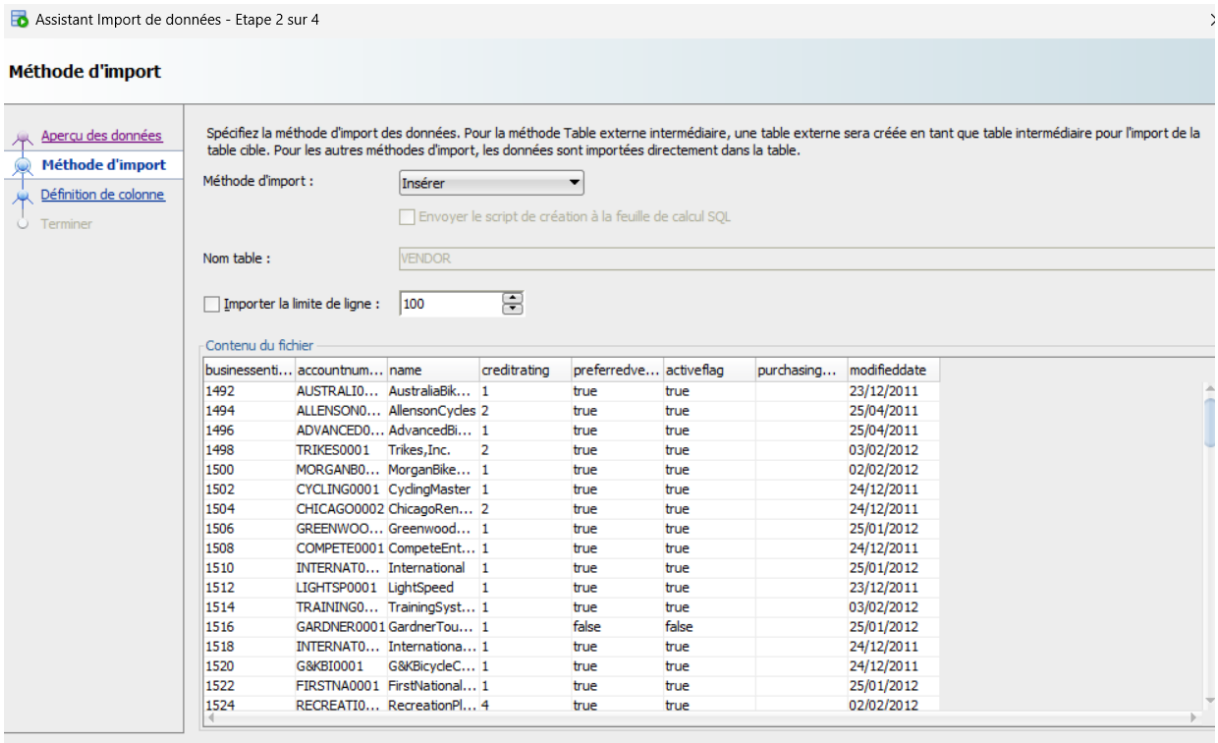
Click on importer les données



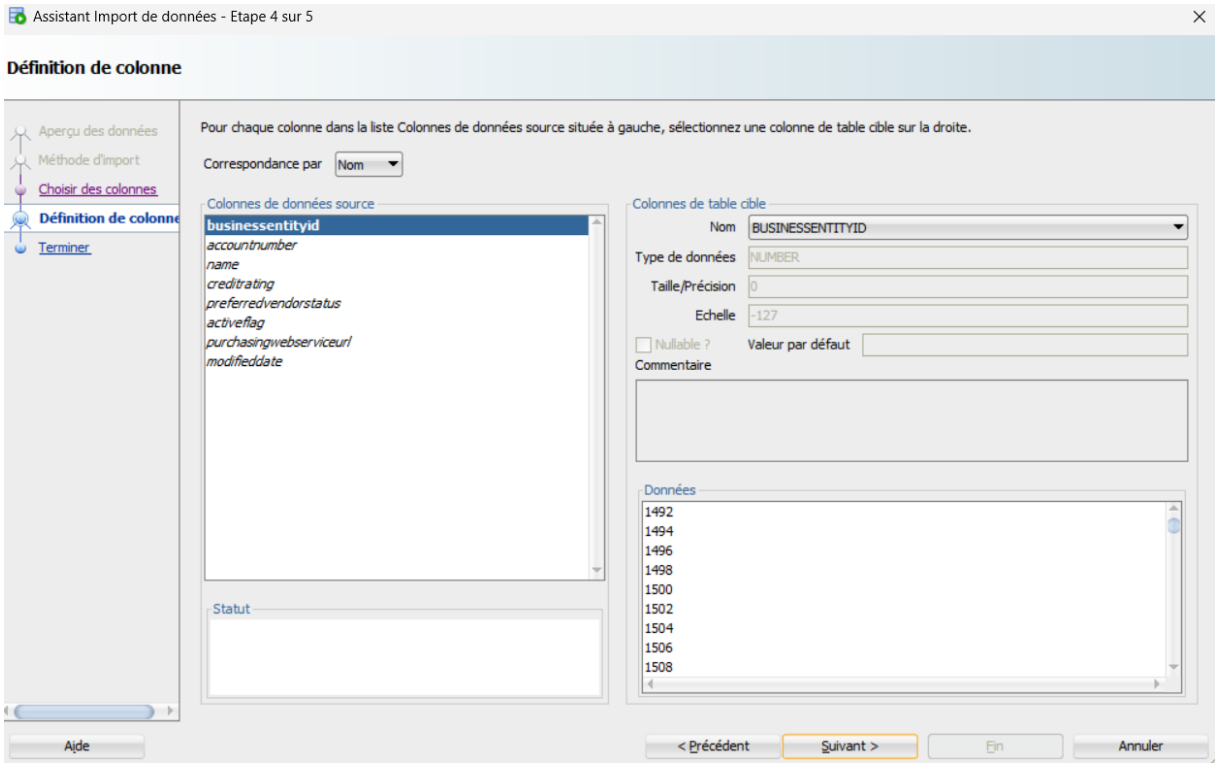
We select « ; » as separator

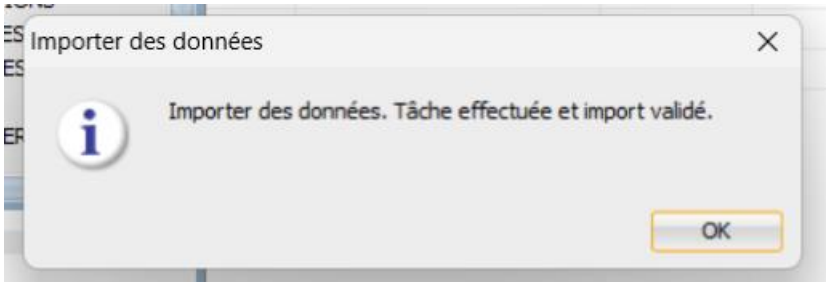


And we click on suivant



Click on suivant





We repeat for the other tables

SQL Query Optimization

Queries were crafted to extract specific data sets, with optimization techniques applied, such as the use of indexes and consideration for execution plans, to enhance performance.

```
--SQL QUERIES
-- A. Vendors with Credit Rating 5 and ProductID > 500
SELECT V.Name, PV.ProductID
FROM Vendor V
JOIN ProductVendor PV ON V.BusinessEntityID = PV.BusinessEntityID
WHERE V.CreditRating = 5 AND PV.ProductID > 500;
```

	A	B
1	NAME	PRODUCTID
2	VictoryBikes	922
3	VictoryBikes	923
4	VictoryBikes	933
5	VictoryBikes	934

```sql
-- B. Purchase Orders with Qty > 500
SELECT POH.PurchaseOrderID, POH.OrderDate, POD.PurchaseOrderDetailID, POD.OrderQty, POD.ProductID
FROM PurchaseOrderHeader POH
JOIN PurchaseOrderDetail POD ON POH.PurchaseOrderID = POD.PurchaseOrderID
WHERE POD.OrderQty > 500;
```

	A	B	C	D	E	F	G
1	PURCHASEOR	ORDERDATE	PURCHASEOR	ORDERQTY	PRODUCTID		
2	3	16/04/2011	4	550	530		
3	5	30/04/2011	6	550	512		
4	6	30/04/2011	7	550	513		
5	7	30/04/2011	8	550	317		
6	7	30/04/2011	9	550	318		
7	7	30/04/2011	10	550	319		
8	12	14/12/2011	28	550	941		

```sql
-- C. Orders 1400 to 1600
SELECT POH.PurchaseOrderID, POH.VendorID, POD.PurchaseOrderDetailID, POD.ProductID, POD.UnitPrice
FROM PurchaseOrderHeader POH
JOIN PurchaseOrderDetail POD ON POH.PurchaseOrderID = POD.PurchaseOrderID
WHERE POH.PurchaseOrderID BETWEEN 1400 AND 1600;
```

	A	B	C	D	E	F	G
1	PURCHASEORDERID,"VENDORID","PURCHASEORDERDETAILID","PRODUCTID","UNITPRICE"						
2	1400,1556,3177,319,465						
3	1401,1554,3178,366,41,223						
4	1401,1554,3179,367,45,4965						
5	1401,1554,3180,368,41,223						
6	1401,1554,3181,369,39,123						
7	1402,1688,3182,2,41,916						
8	1403,1580,3183,1,50,2635						
9	1404,1496,3184,359,47,6805						
10	1404,1496,3185,360,45,5805						
11	1405,1494,3186,530,16,086						
12	1406,1650,3187,4,57,0255						
13	1407,1654,3188,512,37,086						
14	1408,1664,3189,513,26,5965						

```sql
-- D. Orders and Cost per Vendor
SELECT V.BusinessEntityID, COUNT(POH.PurchaseOrderID) AS NumberOfOrders, SUM(POD.UnitPrice * POD.OrderQty) AS TotalCost
FROM Vendor V
JOIN PurchaseOrderHeader POH ON V.BusinessEntityID = POH.VendorID
JOIN PurchaseOrderDetail POD ON POH.PurchaseOrderID = POD.PurchaseOrderID
GROUP BY V.BusinessEntityID
ORDER BY TotalCost DESC;
```

	A	B	C	D
1	BUSINESSENTITYID,"NUMBEROFORDERS","TOTALCOST"			
2	1556,50,12787500			
3	1576,100,4555897,5			
4	1684,142,3058774,95			
5	1696,179,3029108,775			
6	1680,120,2553243			
7	1578,120,2513742			

```sql
-- E. Average Orders and Cost Across Vendors
SELECT AVG(NumberOfOrders) AS AvgOrders, AVG(TotalCost) AS AvgCost
FROM (
    SELECT V.BusinessEntityID, COUNT(POH.PurchaseOrderID) AS NumberOfOrders, SUM(POD.UnitPrice * POD.OrderQty) AS TotalCost
    FROM Vendor V
    JOIN PurchaseOrderHeader POH ON V.BusinessEntityID = POH.VendorID
    JOIN PurchaseOrderDetail POD ON POH.PurchaseOrderID = POD.PurchaseOrderID
    GROUP BY V.BusinessEntityID
) VendorOrders;
```

	A	B
1	AVGORDERS	AVGCOST
2	102,85	883841,83

```sql
-- F. Top 10 Vendors by Rejected Items Percentage
SELECT V.BusinessEntityID, (SUM(POD.RejectedQty) / SUM(POD.ReceivedQty)) * 100 AS RejectionPercentage
FROM Vendor V
JOIN PurchaseOrderHeader POH ON V.BusinessEntityID = POH.VendorID
JOIN PurchaseOrderDetail POD ON POH.PurchaseOrderID = POD.PurchaseOrderID
GROUP BY V.BusinessEntityID
ORDER BY RejectionPercentage DESC
FETCH FIRST 10 ROWS ONLY;
```

	A	B	C
1	BUSINESSENT	REJECTIONPERCENTAGE	
2	1510	5,88	
3	1664	5,48	
4	1588	5,4	
5	1658	5,33	
6	1560	5,33	
7	1586	5,29	
8	1682	5,27	
9	1576	5,25	
10	1620	5,17	
11	1590	5,15	

```sql
-- G. Top 10 Vendors by Largest Orders
SELECT V.BusinessEntityID, SUM(POD.OrderQty) AS TotalQuantity
FROM Vendor V
JOIN PurchaseOrderHeader POH ON V.BusinessEntityID = POH.VendorID
JOIN PurchaseOrderDetail POD ON POH.PurchaseOrderID = POD.PurchaseOrderID
GROUP BY V.BusinessEntityID
ORDER BY TotalQuantity DESC
FETCH FIRST 10 ROWS ONLY;
```

	A	B
1	BUSINESSENT	TOTALQUANTITY
2	1590	125
3	1568	1155
4	1696	9845
5	1652	792
6	1684	781
7	1544	693
8	1508	693
9	1694	6875
10	1570	6765
11	1646	671

```
-- H. Top 10 Products by Quantity Purchased
SELECT POD.ProductID, SUM(POD.OrderQty) AS TotalQuantity
FROM PurchaseOrderDetail POD
GROUP BY POD.ProductID
ORDER BY TotalQuantity DESC
FETCH FIRST 10 ROWS ONLY;
```

	A	B
1	PRODUCTID	TOTALQUANTITY
2	319	715
3	325	625
4	326	625
5	507	561
6	508	561
7	524	561
8	523	561
9	936	561
10	935	561
11	513	5555

```
-- I. Complex Queries with Analytic Functions (Example)
-- Example query to illustrate the use of analytic functions
SELECT ProductID, SUM(OrderQty) OVER (PARTITION BY ProductID) AS TotalOrdersPerProduct
FROM PurchaseOrderDetail
ORDER BY ProductID;
```

	A	B	C
1	PRODUCTID	TOTALORDERSPERPRODUCT	
2	1	154	
3	1	154	
4	1	154	
5	1	154	
6	1	154	
7	1	154	

We download all the outputs of the queries in csv files.

Trigger Implementation

Triggers **BEFORE_UPDATE_POD** and **BEFORE_UPDATE_POH** were implemented to maintain transaction history and ensure subtotal consistency. Challenges encountered with trigger compilation were resolved, ensuring triggers operated correctly.

```
-- Triggers
-- J. Trigger for Transaction_History and PurchaseOrderDetail Updates
CREATE OR REPLACE TRIGGER Before_Update_POD
BEFORE UPDATE ON PurchaseOrderDetail
FOR EACH ROW
BEGIN
    INSERT INTO Transaction_History VALUES (:NEW.PurchaseOrderID, :NEW.PurchaseOrderDetailID,
    UPDATE PurchaseOrderDetail
    SET ModifiedDate = SYSDATE
    WHERE PurchaseOrderDetailID = :NEW.PurchaseOrderDetailID;
END;
```

```
-- K. Trigger to Ensure SubTotal Consistency
CREATE OR REPLACE TRIGGER Before_Update_POH
BEFORE UPDATE OF Subtotal ON PurchaseOrderHeader
FOR EACH ROW
DECLARE
    TotalDetailAmount NUMBER;
BEGIN
    SELECT SUM(UnitPrice * OrderQty) INTO TotalDetailAmount
    FROM PurchaseOrderDetail
    WHERE PurchaseOrderID = :NEW.PurchaseOrderID;
    IF TotalDetailAmount != :NEW.Subtotal THEN
        RAISE_APPLICATION_ERROR(-20001, 'Subtotal not consistent with PurchaseOrderDetail da
    END IF;
END;
```

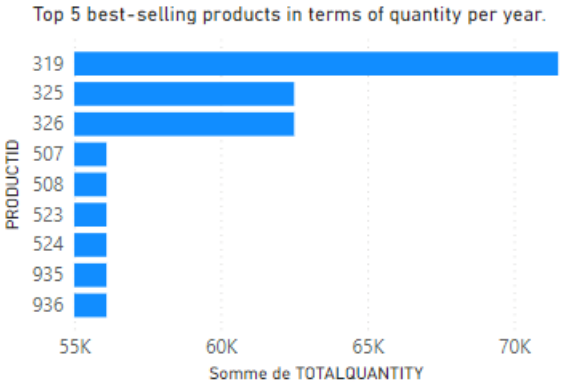
Data Export Process

A procedural methodology was followed to export query results from Oracle SQL Developer to CSV files, which served as the data foundation for Power BI visualizations.

Power BI Dashboard Development

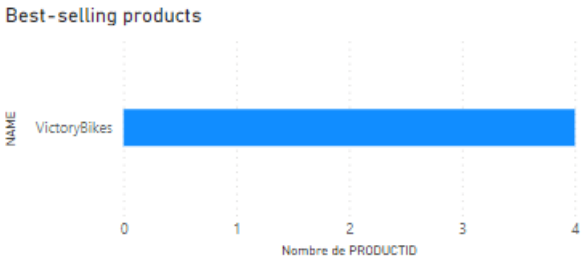
Dashboards were created to display data such as the top-selling products and vendors with the lowest rejection rates. Visualizations were selected and optimized for clarity and interpretability.

```
-- Top 5 Vendors by Lowest Rejected Items Percentage
SELECT V.BusinessEntityID,
       (SUM(POD.RejectedQty) / NULLIF(SUM(POD.ReceivedQty), 0)) * 100 AS RejectionPercentage
FROM Vendor V
JOIN PurchaseOrderHeader POH ON V.BusinessEntityID = POH.VendorID
JOIN PurchaseOrderDetail POD ON POH.PurchaseOrderID = POD.PurchaseOrderID
GROUP BY V.BusinessEntityID
HAVING SUM(POD.ReceivedQty) > 0  -- To avoid division by zero
ORDER BY RejectionPercentage ASC
FETCH FIRST 5 ROWS ONLY;
```



Top 5 vendors who have the lowest percentage of rejected received items

BUSINESSENTITYID	Somme de REJECTIONPERCENTAGE
1536	0
1592	0
1604	0
1618	0
1648	0
Total	**0**



Analysis and Interpretation

The Power BI dashboards synthesized complex datasets into strategic insights, showcasing pivotal sales trends and supply chain efficiencies. The visualizations not only pinpointed the leading products driving the market but also shed light on the vendors exemplifying optimal performance through minimal rejection rates. This analytical narrative allowed us to discern market demands and supply consistency, facilitating data-driven decisions for inventory management and quality control. These insights underscore a path forward for strategic planning and competitive positioning in the market.

Conclusion

In conclusion, this report has navigated the challenges and complexities of advanced database management and data visualization. It has documented the process of creating a robust SQL database, optimizing queries for performance, and confronting the intricacies of trigger functionalities. The journey through exporting data and crafting insightful Power BI dashboards has underscored the significance of precision and adaptability in handling and presenting data. These experiences have equipped us with valuable lessons for enhancing future data-driven strategies and underscored the importance of resilience and continuous learning in the face of technical hurdles. This foundation paves the way for advanced analytics and strategic business intelligence operations, where data is not only a resource but a beacon guiding decision-making processes.