SQL Queries Explained

Database Setup

1. Create Database and Use It

```
CREATE DATABASE adidas_sales_db;
USE adidas_sales_db;
```

Explanation: Creates a new database named `adidas_sales_db` and selects it for use.

2. Rename Table for Simplicity

```
RENAME TABLE `adidas us sales datasets` TO adidas;
```

Explanation: Renames the table to `adidas` for easier reference in queries.

Data Cleaning and Transformation

3. Check Data

```
SELECT * FROM adidas;
SHOW COLUMNS FROM adidas;
```

Explanation: Displays all records in the `adidas` table and shows the column structure.

4. Convert `Invoice Date` to DATE Format

```
UPDATE adidas SET `Invoice Date` = STR_TO_DATE(`Invoice Date`, '%m/%d/%Y');
ALTER TABLE adidas MODIFY `Invoice Date` DATE;
```

Explanation: Converts the `Invoice Date` column from string format to a proper DATE format.

5. Convert `Price per Unit` to INT

```
UPDATE adidas SET `Price per Unit` = REPLACE(`Price per Unit`, '$', '');
ALTER TABLE adidas MODIFY `Price per Unit` INT;
```

Explanation: Removes the `$` sign and changes the column data type to INT.

6. Convert `Units Sold` to INT

```
UPDATE adidas SET `Units Sold` = REPLACE(`Units Sold`, ',', '');
DELETE FROM adidas WHERE `Units Sold` = 0;
ALTER TABLE adidas MODIFY `Units Sold` INT;
```

Explanation: Removes commas, deletes zero unit sales, and converts the column to INT.

7. Convert `Total Sales` and `Operating Profit` to INT

```
UPDATE adidas SET `Total Sales` = REPLACE(REPLACE(`Total Sales`, '$', ''),
',', '');
ALTER TABLE adidas MODIFY `Total Sales` INT;

UPDATE adidas SET `Operating Profit` = REPLACE(REPLACE(`Operating Profit`,
'$', ''), ',', '');
ALTER TABLE adidas MODIFY `Operating Profit` INT;
```

Explanation: Removes currency symbols, formats the values properly, and changes the data type.

8. Convert `Operating Margin` to INT

```
UPDATE adidas SET `Operating Margin` = REPLACE(`Operating Margin`, '%', '');
ALTER TABLE adidas MODIFY `Operating Margin` INT;
```

Explanation: Removes percentage signs and converts the column to INT.

9. Check for Missing Values

```
SELECT COUNT(*) FROM adidas;
SELECT * FROM adidas WHERE `Retailer` IS NULL;
SELECT * FROM adidas WHERE `Retailer ID` IS NULL OR `Retailer ID` = 0;
SELECT * FROM adidas WHERE `Invoice Date` IS NULL;
```

Explanation: Identifies missing values in key columns.

10. Check for Duplicates

```
SELECT `Invoice Date`, `Retailer ID`, `Product`, COUNT(*)
FROM adidas
GROUP BY `Invoice Date`, `Retailer ID`, `Product`
HAVING COUNT(*) > 1;
```

Explanation: Finds duplicate rows based on key identifiers.

11. Add and Update Category Column

```
ALTER TABLE adidas ADD COLUMN category VARCHAR(255) AFTER city;
UPDATE adidas SET category =
CASE
    WHEN `Product` LIKE "%Footwear%" THEN "footwear"
    WHEN `Product` LIKE "%Apparel%" THEN "apparel"
    ELSE "other"
END;
```

Explanation: Adds a `category` column and classifies products into categories.

12. Add and Update Gender Column

```
ALTER TABLE adidas ADD COLUMN gender VARCHAR(255) AFTER `Product`;
UPDATE adidas SET gender =
CASE
    WHEN `Product` LIKE "Men%" THEN "male"
    WHEN `Product` LIKE "Women%" THEN "female"
    ELSE "other"
END;
```

Explanation: Adds a `gender` column and classifies products by gender.

13. Add Month Column

```
ALTER TABLE adidas ADD COLUMN `month` VARCHAR(255) AFTER `Invoice Date`;
UPDATE adidas SET `month` = MONTHNAME(`Invoice Date`);
```

Explanation: Adds a column for the month of each transaction.

Sales Performance Insights

14. Total Revenue Generated

```
SELECT SUM(`Total Sales`) AS `Total Sales` FROM adidas;
```

Explanation: Calculates the total revenue.

15. Top 5 States and Cities by Sales

```
SELECT state, city, SUM(`Total Sales`) AS total_sales
FROM adidas
GROUP BY state, city
ORDER BY total_sales DESC LIMIT 5;
```

Explanation: Identifies the highest revenue-generating locations.

16. Revenue by Product Category

```
SELECT category, SUM(`Total Sales`) AS total_sales
FROM adidas
GROUP BY category
ORDER BY total_sales DESC;
```

Explanation: Compares total sales across categories.

17. Highest Units Sold by Category

```
SELECT category, SUM(`Units Sold`) AS total_units_sold
FROM adidas
GROUP BY category
ORDER BY total_units_sold DESC;
```

Explanation: Finds the category with the highest sales volume.

18. Highest Operating Profit by Category

```
SELECT category, SUM(`Operating Profit`) AS total_operating_profit
FROM adidas
GROUP BY category
ORDER BY total_operating_profit DESC;
```

Explanation: Identifies the most profitable category.

Gender-Based Insights

19. Total Sales by Gender

```
SELECT gender, SUM(`Total Sales`) AS total_sales
FROM adidas
GROUP BY gender
ORDER BY total_sales DESC;
```

Explanation: Compares sales performance between male and female products.

20. Best-Selling Product for Each Gender

```
SELECT gender, `Product`, SUM(`Total Sales`) AS `Total Sales`
FROM (
    SELECT gender, `Product`, SUM(`Total Sales`) AS `Total Sales`,
           RANK() OVER(PARTITION BY gender ORDER BY SUM(`Total Sales`) DESC)
AS rnk
    FROM adidas
    GROUP BY gender, `Product`
) ranked_product
WHERE rnk = 1;
```

Explanation: Finds the best-selling product in each gender category.

Time-Based Insights

21. Best and Worst Revenue Quarters

```
SELECT QUARTER(`Invoice Date`) AS quarter, SUM(`Total Sales`) AS total_sales
FROM adidas
GROUP BY quarter
ORDER BY total_sales DESC LIMIT 1;
```

Explanation: Identifies the best revenue quarter.

```
SELECT QUARTER(`Invoice Date`) AS quarter, SUM(`Total Sales`) AS total_sales
FROM adidas
GROUP BY quarter
ORDER BY total_sales ASC LIMIT 1;
```

Explanation: Identifies the worst revenue quarter.

22. Best Average Operating Margin by Retailer

```
SELECT retailer, AVG(`Operating Margin`) AS avg_margin
FROM adidas
GROUP BY retailer
ORDER BY avg_margin DESC LIMIT 1;
```

Explanation: This query calculates the average operating margin for each retailer and finds the retailer with the highest average operating margin.

---

23. Top 5 Retailers by Total Sales

```
SELECT retailer, SUM(`Total Sales`) AS total_sales
FROM adidas
GROUP BY retailer
ORDER BY total_sales DESC LIMIT 5;
```

Explanation: This query aggregates the total sales for each retailer and displays the top 5 retailers with the highest sales.

---

24. Top 5 Retailers by Units Sold

```
SELECT retailer, SUM(`Units Sold`) AS total_units_sold
FROM adidas
GROUP BY retailer
ORDER BY total_units_sold DESC LIMIT 5;
```

Explanation: This query finds the top 5 retailers who sold the highest number of units by summing the `Units Sold` column.

---

25. Retailer Performance by Category

```
SELECT retailer, category, SUM(`Total Sales`) AS total_sales
FROM adidas
```

```
GROUP BY retailer, category
ORDER BY retailer, total_sales DESC;
```

Explanation: This query groups sales data by retailer and category, showing how each retailer performed in different product categories.

---

26. Total Number of Units Sold Overall

```
SELECT SUM(`Units Sold`) AS Total_Units_Sold FROM adidas;
```

Explanation: This query calculates the total number of units sold across all transactions in the dataset.

---

27. Product with the Highest Total Units Sold

```
SELECT `Product`, SUM(`Units Sold`) AS Total_Units_Sold
FROM adidas
GROUP BY `Product`
ORDER BY Total_Units_Sold DESC
LIMIT 1;
```

Explanation: This query determines the product that sold the highest number of units.

---

28. Highest Units Sold by Product Category

```
SELECT `Category`, SUM(`Units Sold`) AS Total_Units_Sold
FROM adidas
GROUP BY `Category`
ORDER BY Total_Units_Sold DESC;
```

Explanation: This query calculates and compares the total units sold for each product category.

---

29. State with the Highest Number of Units Sold

```
SELECT `State`, SUM(`Units Sold`) AS Total_Units_Sold
FROM adidas
GROUP BY `State`
ORDER BY Total_Units_Sold DESC
LIMIT 1;
```

Explanation: This query finds the state that had the highest number of units sold.

30. Retailer with the Highest Average Units Sold per Order

```
SELECT `Retailer`, AVG(`Units Sold`) AS Avg_Units_Sold
FROM adidas
GROUP BY `Retailer`
ORDER BY Avg_Units_Sold DESC
LIMIT 1;
```

Explanation: This query finds the retailer with the highest average units sold per order.

---

31. Trend of Total Units Sold Per Month

```
SELECT `Month`, SUM(`Units Sold`) AS Total_Units_Sold
FROM adidas
GROUP BY `Month`
ORDER BY `Month`;
```

Explanation: This query examines the total units sold for each month to identify trends.

---

32. Month with the Highest Units Sold

```
SELECT `Month`, SUM(`Units Sold`) AS Total_Units_Sold
FROM adidas
GROUP BY `Month`
ORDER BY Total_Units_Sold DESC
LIMIT 1;
```

Explanation: This query determines which month had the highest number of units sold.

---

33. Top 5 Best-Selling Products by Revenue

```
SELECT `Product`, SUM(`Total Sales`) AS Total_Revenue
FROM adidas
GROUP BY `Product`
ORDER BY Total_Revenue DESC
LIMIT 5;
```

Explanation: This query identifies the top 5 best-selling products based on total revenue.

---

34. Least Profitable Products

```
SELECT `Product`, SUM(`Operating Profit`) AS Total_Profit
FROM adidas
GROUP BY `Product`
ORDER BY Total_Profit ASC
LIMIT 5;
```

Explanation: This query finds the 5 products with the lowest operating profit.

---

35. Monthly Sales Performance

```
SELECT `Month`, SUM(`Total Sales`) AS Total_Revenue
FROM adidas
GROUP BY `Month`
ORDER BY `Month`;
```

Explanation: This query shows the total sales for each month in chronological order.

---

36. Top 5 Cities with Highest Sales

```
SELECT `City`, SUM(`Total Sales`) AS Total_Revenue
FROM adidas
GROUP BY `City`
ORDER BY Total_Revenue DESC
LIMIT 5;
```

Explanation: This query identifies the top 5 cities with the highest sales.

---

37. Products with the Highest Units Sold

```
SELECT `Product`, SUM(`Units Sold`) AS Total_Units
FROM adidas
GROUP BY `Product`
ORDER BY Total_Units DESC
LIMIT 5;
```

Explanation: This query finds the top 5 products with the highest units sold.

---

38. Compare Sales Method Performance

```
SELECT `Sales Method`, SUM(`Total Sales`) AS Total_Revenue
FROM adidas
GROUP BY `Sales Method`;
```

Explanation: This query compares different sales methods based on total revenue generated.