SQL Queries and Database Setup for Sales and Products

Database Tables Setup

```sql
-- Products Table
CREATE TABLE Products (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    product_name VARCHAR(100) NOT NULL,
    category VARCHAR(50),
    unit_price DECIMAL(10,2) NOT NULL
);

-- Sales Table
CREATE TABLE Sales (
    sale_id INT AUTO_INCREMENT PRIMARY KEY,
    product_id INT,
    sale_date DATE,
    quantity_sold INT,
    total_price DECIMAL(10,2),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);

-- Insert Sample Data
INSERT INTO Products (product_name, category, unit_price) VALUES
('Smartphone X', 'Electronics', 500.00),
('Laptop Pro', 'Electronics', 1200.00),
('Headphones', 'Electronics', 150.00),
('Office Chair', 'Furniture', 200.00),
('Desk Lamp', 'Furniture', 50.00),
('Gaming Mouse', 'Electronics', 80.00),
('Notebook', 'Stationery', 5.00),
('Pen', 'Stationery', 2.00);

INSERT INTO Sales (product_id, sale_date, quantity_sold, total_price) VALUES
(1, '2024-01-01', 2, 1000.00),
(2, '2024-01-01', 1, 1200.00),
(3, '2024-01-03', 3, 450.00),
(1, '2024-01-03', 1, 500.00),
(4, '2024-01-04', 5, 1000.00),
(5, '2024-01-05', 2, 100.00),
(6, '2024-01-06', 4, 320.00),
(7, '2024-01-06', 10, 50.00);
```

Basic Queries

1. Retrieve all columns from the Sales table.

```sql
SELECT * FROM Sales;
```

2. Retrieve product_name and unit_price from Products.

```sql
SELECT product_name, unit_price FROM Products;
```

3. Retrieve sale_id and sale_date from Sales.

```sql
SELECT sale_id, sale_date FROM Sales;
```

4. Sales with total_price > 100.

```sql
SELECT * FROM Sales WHERE total_price > 100;
```

5. Products in 'Electronics' category.

```sql
SELECT * FROM Products WHERE category = 'Electronics';
```

6. Sales on '2024-01-03'.

```sql
SELECT sale_id, total_price FROM Sales WHERE sale_date = '2024-01-03';
```

7. Products with unit_price > 100.

```sql
SELECT product_id, product_name FROM Products WHERE unit_price > 100;
```

8. Total revenue.

```sql
SELECT SUM(total_price) AS total_revenue FROM Sales;
```

9. Average unit_price.

```sql
SELECT AVG(unit_price) AS average_unit_price FROM Products;
```

10. Total quantity_sold.

```sql
SELECT SUM(quantity_sold) AS total_quantity_sold FROM Sales;
```

11. Sales per day.

```sql
SELECT sale_date, COUNT(*) AS sales_count FROM Sales GROUP BY sale_date
ORDER BY sale_date;
```

12. Product with highest unit_price.

```sql
SELECT product_name, unit_price FROM Products ORDER BY unit_price DESC
LIMIT 1;
```

13. Sales with quantity_sold > 4.

```sql
SELECT sale_id, product_id, total_price FROM Sales WHERE quantity_sold > 4;
```

14. Products ordered by unit_price descending.

```sql
SELECT product_name, unit_price FROM Products ORDER BY unit_price DESC;
```

15. Total sales rounded.

```sql
SELECT ROUND(SUM(total_price), 2) AS total_sales FROM Sales;
```

16. Average total_price.

```sql
SELECT AVG(total_price) AS average_total_price FROM Sales;
```

17. Format sale_date.

```
SELECT sale_id, DATE_FORMAT(sale_date, '%Y-%m-%d') AS formatted_date FROM
Sales;
```

18. Total revenue for 'Electronics'.

```
SELECT SUM(Sales.total_price) AS total_revenue
FROM Sales
JOIN Products ON Sales.product_id = Products.product_id
WHERE Products.category = 'Electronics';
```

19. Products with unit_price between 20 and 600.

```
SELECT product_name, unit_price FROM Products WHERE unit_price BETWEEN 20
AND 600;
```

20. Products ordered by category.

```
SELECT product_name, category FROM Products ORDER BY category ASC;
```

Intermediate Queries

1. Total quantity_sold for 'Electronics'.

```
SELECT SUM(quantity_sold) AS total_quantity_sold
FROM Sales
JOIN Products ON Sales.product_id = Products.product_id
WHERE Products.category = 'Electronics';
```

2. product_name and total_price (quantity_sold * unit_price).

```
SELECT product_name, quantity_sold * unit_price AS total_price
FROM Sales
JOIN Products ON Sales.product_id = Products.product_id;
```

3. Most frequently sold product.

```sql
SELECT product_id, COUNT(*) AS sales_count
FROM Sales
GROUP BY product_id
ORDER BY sales_count DESC
LIMIT 1;
```

4. Products not sold.

```sql
SELECT product_id, product_name
FROM Products
WHERE product_id NOT IN (SELECT DISTINCT product_id FROM Sales);
```

5. Total revenue per category.

```sql
SELECT p.category, SUM(s.total_price) AS total_revenue
FROM Sales s
JOIN Products p ON s.product_id = p.product_id
GROUP BY p.category;
```

6. Category with highest average unit_price.

```sql
SELECT category
FROM Products
GROUP BY category
ORDER BY AVG(unit_price) DESC
LIMIT 1;
```

7. Products with total sales > 30.

```sql
SELECT p.product_name
FROM Sales s
JOIN Products p ON s.product_id = p.product_id
GROUP BY p.product_name
HAVING SUM(s.total_price) > 30;
```

8. Number of sales per month.

```sql
SELECT DATE_FORMAT(s.sale_date, '%Y-%m') AS month, COUNT(*) AS sales_count
FROM Sales s
GROUP BY month;
```

9. Sales for products with 'Smart' in name.

```sql
SELECT s.sale_id, p.product_name, s.total_price
FROM Sales s
JOIN Products p ON s.product_id = p.product_id
WHERE p.product_name LIKE '%Smart%';
```

10. Average quantity sold for products with unit_price > 100.

```sql
SELECT AVG(s.quantity_sold) AS average_quantity_sold
FROM Sales s
JOIN Products p ON s.product_id = p.product_id
WHERE p.unit_price > 100;
```

11. Product name and total revenue.

```sql
SELECT p.product_name, SUM(s.total_price) AS total_revenue
FROM Sales s
JOIN Products p ON s.product_id = p.product_id
GROUP BY p.product_name;
```

12. List all sales with product names.

```sql
SELECT s.sale_id, p.product_name
FROM Sales s
JOIN Products p ON s.product_id = p.product_id;
```

13. Category revenue and percentage.

```sql
SELECT p.category,
       SUM(s.total_price) AS category_revenue,
       (SUM(s.total_price) / (SELECT SUM(total_price) FROM Sales)) * 100
AS revenue_percentage
FROM Sales s
```

```
JOIN Products p ON s.product_id = p.product_id
GROUP BY p.category
ORDER BY revenue_percentage DESC
LIMIT 3;
```

14. Days since each sale.

```
SELECT sale_id, DATEDIFF(NOW(), sale_date) AS days_since_sale
FROM Sales;
```