here are some examples of how you could use these basic SQL techniques for financial analysis:

1. Aggregate functions: you can use aggregate functions to calculate basic financial metrics such as total revenue or average cost. For example, the following query calculates the total revenue for a given product:

```

SELECT SUM(Quantity \* Price) AS Revenue

FROM Sales

WHERE ProductID = @ProductID

```

2. Group by: you can use the GROUP BY clause to group financial data by dimensions such as time, geography, or product lines. For example, the following query calculates the total revenue by quarter:

```

SELECT YEAR(OrderDate) AS Year,

QUARTER(OrderDate) AS Quarter,

SUM(Quantity \* Price) AS Revenue

FROM Sales

GROUP BY YEAR(OrderDate), QUARTER(OrderDate)

```

3. Join: you can use joins to combine financial data from multiple tables, such as sales data and product information. For example, the following query retrieves all sales for a given product, along with its name and cost:

```

SELECT Sales.OrderID, Sales.Quantity, Sales.Price, Products.ProductName, Products.UnitCost

FROM Sales

INNER JOIN Products ON Sales.ProductID = Products.ProductID

WHERE Sales.ProductID = @ProductID

```

4. Subqueries: you can use subqueries to filter financial data based on criteria such as maximum or minimum values. For example, the following query retrieves all sales for the most expensive product:

```

SELECT Sales.OrderID, Sales.Quantity, Sales.Price, Products.ProductName

FROM Sales

INNER JOIN Products ON Sales.ProductID = Products.ProductID

WHERE Products.UnitPrice = (SELECT MAX(UnitPrice) FROM Products)

```

5. Order by: you can use the ORDER BY clause to sort financial data by metrics such as revenue, profit, or growth rate. For example, the following query retrieves the top 10 products by revenue:

```

SELECT Products.ProductName, SUM(Sales.Quantity \* Sales.Price) AS Revenue

FROM Sales

INNER JOIN Products ON Sales.ProductID = Products.ProductID

GROUP BY Products.ProductName

ORDER BY Revenue DESC

LIMIT 10

```

6. Limit: you can use the LIMIT clause to restrict the number of rows returned by a query, which can be useful for financial reports or dashboards. For example, the following query retrieves the top 5 salespeople by revenue:

```

SELECT Employees.FirstName, Employees.LastName, SUM(Sales.Quantity \* Sales.Price) AS Revenue

FROM Sales

INNER JOIN Employees ON Sales.EmployeeID = Employees.EmployeeID

GROUP BY Employees.FirstName, Employees.LastName

ORDER BY Revenue DESC

LIMIT 5

```

Here are some examples of how you could use these advanced SQL techniques for intermediate level financial analysis:

1. Window functions: you can use window functions to calculate financial ratios such as price-to-earnings (P/E) or price-to-book (P/B) ratios. For example, the following query calculates the P/E ratio for each stock in a given portfolio:

```

WITH EarningsCTE AS (

SELECT StockID, SUM(Earnings) AS TotalEarnings

FROM QuarterlyReports

GROUP BY StockID

)

SELECT StockID, Price / (EarningsCTE.TotalEarnings / 4) AS PE\_Ratio

FROM Stocks

INNER JOIN EarningsCTE ON Stocks.StockID = EarningsCTE.StockID

```

2. Common Table Expressions (CTEs): you can use CTEs to calculate financial metrics such as return on investment (ROI) or net present value (NPV). For example, the following query calculates the NPV for a given investment at a discount rate of 10%:

```

WITH CashFlowsCTE AS (

SELECT InvestmentID, CashFlow / POWER(1.1, Year - 1) AS DiscountedCashFlow

FROM CashFlows

WHERE InvestmentID = @InvestmentID

)

SELECT SUM(DiscountedCashFlow) AS NPV

FROM CashFlowsCTE

```

3. Recursive queries: you can use recursive queries to traverse financial hierarchies such as account structures or financial statements. For example, the following query returns all accounts and their subaccounts in a hierarchical format:

```

WITH AccountHierarchy AS (

SELECT AccountID, AccountName, ParentID, 0 AS Level

FROM Accounts

WHERE ParentID IS NULL

UNION ALL

SELECT Accounts.AccountID, Accounts.AccountName, Accounts.ParentID, AccountHierarchy.Level + 1

FROM Accounts

INNER JOIN AccountHierarchy ON Accounts.ParentID = AccountHierarchy.AccountID

)

SELECT REPLICATE('-', Level) + AccountName AS AccountPath, Balance

FROM AccountHierarchy

LEFT JOIN Balances ON AccountHierarchy.AccountID = Balances.AccountID

```

4. Stored procedures: you can use stored procedures to automate financial workflows such as budgeting or forecasting. For example, the following stored procedure calculates the net income and generates an alert if it falls below a certain threshold:

```

CREATE PROCEDURE CalculateNetIncome

@Threshold DECIMAL(18,2)

AS

BEGIN

DECLARE @NetIncome DECIMAL(18,2)

SET @NetIncome = (SELECT SUM(Revenue) - SUM(Expenses) FROM IncomeStatement)

IF @NetIncome < @Threshold

BEGIN

EXEC sp\_send\_dbmail @recipients='finance@company.com', @subject='Net Income Alert', @body='Net income is below threshold'

END

END

```

5. Indexes: you can use indexes to improve the performance of financial queries that involve complex joins or aggregations. For example, the following index improves the performance of a query that retrieves all transactions for a given account:

```

CREATE INDEX idx\_transactions\_account ON Transactions (AccountID)

```

6. Pivot tables: you can use pivot tables to summarize financial data and compare metrics across dimensions such as time, product lines, or geographies. For example, the following query summarizes the revenue and profit by product line and quarter:

```

SELECT ProductLine,

SUM(CASE WHEN Quarter = 'Q1' THEN Revenue ELSE 0 END) AS 'Q1 Revenue',

SUM(CASE WHEN Quarter = 'Q1' THEN Profit ELSE 0 END) AS 'Q1 Profit',

SUM(CASE WHEN Quarter = 'Q2' THEN Revenue ELSE 0 END) AS 'Q2 Revenue',

SUM(CASE WHEN Quarter = 'Q2' THEN Profit ELSE 0 END) AS 'Q2 Profit',

SUM(CASE WHEN Quarter = 'Q3' THEN Revenue ELSE 0 END) AS 'Q3 Revenue',

SUM(CASE WHEN Quarter = 'Q3' THEN Profit ELSE 0 END) AS 'Q3 Profit',

SUM(CASE WHEN Quarter = 'Q4' THEN Revenue ELSE 0 END) AS 'Q4 Revenue',

SUM(CASE WHEN Quarter = 'Q4' THEN Profit ELSE 0 END) AS 'Q4 Profit'

FROM Sales

INNER JOIN Products ON Sales.ProductID = Products.ProductID

GROUP BY ProductLine

```