**SECTION 1: ANALYSIS OF EACH TABLE  
For each table, I have created advanced SQL queries to extract key insights.**

**1. Employees Table Analysis**

--1.1 Calculate the average salary in each department and analyze gender-based ---salary disparities.  
  
 SELECT department,

AVG(salary) AS Average\_Salary,

AVG(CASE WHEN gender = 'M' THEN salary ELSE NULL END) AS Avg\_Male\_Salary,

AVG(CASE WHEN gender = 'F' THEN salary ELSE NULL END) AS Avg\_Female\_Salary,

(AVG(CASE WHEN gender = 'M' THEN salary ELSE NULL END) - AVG(CASE WHEN gender = 'F' THEN salary ELSE NULL END)) AS Gender\_Pay\_Gap

FROM employees

GROUP BY department;  
  
-- 1.2 Assess the average tenure of employees and calculate the turnover rate.  
 SELECT department,

AVG(DATEDIFF(CURRENT\_DATE, hire\_date)) / 365 AS Avg\_Tenure\_Years,

COUNT(\*) AS Total\_Employees,

SUM(CASE WHEN hire\_date <= DATE\_SUB(CURRENT\_DATE, INTERVAL 1 YEAR) THEN 1 ELSE 0 END) / COUNT(\*) \* 100 AS Turnover\_Rate\_Percent

FROM employees

GROUP BY department;  
  
-- 1.3 Identify top-performing employees based on sales contribution.  
  
 SELECT e.employee\_id, e.first\_name, e.department, SUM(s.Sales) AS Total\_Sales

FROM employees e

JOIN sales s ON e.employee\_id = s.Customer\_ID

GROUP BY e.employee\_id, e.first\_name, e.department

ORDER BY Total\_Sales DESC

LIMIT 10;  
  
-- 1.4 Analyze the distribution of employees across different regions.  
 SELECT region\_id, COUNT(\*) AS Employee\_Count

FROM employees

GROUP BY region\_id;

-- 1.5 Examine the distribution of salaries and identify outliers.

SELECT department, salary,

PERCENTILE\_CONT(0.5) WITHIN GROUP (ORDER BY salary) OVER (PARTITION BY department) AS Median\_Salary,

PERCENTILE\_CONT(0.25) WITHIN GROUP (ORDER BY salary) OVER (PARTITION BY department) AS Lower\_Quartile,

PERCENTILE\_CONT(0.75) WITHIN GROUP (ORDER BY salary) OVER (PARTITION BY department) AS Upper\_Quartile

FROM employees;  
  
  
**2. Departments Table Analysis**

--2.1 Determine the size of each department and its division.

SELECT d.department, d.division, COUNT(e.employee\_id) AS Employee\_Count

FROM departments d

LEFT JOIN employees e ON d.department = e.department

GROUP BY d.department, d.division;  
  
-- 2.2 Identify departments within the same division to facilitate

--collaboration.  
  
 SELECT DISTINCT d1.department AS Department1, d2.department AS Department2, d1.division

FROM departments d1

JOIN departments d2 ON d1.division = d2.division AND d1.department != d2.department;  
  
  
**3. Regions Table Analysis**-- 3.1 Analyze the distribution of employees and their sales performance by   
--region.

SELECT r.region\_id, r.region, r.country,

COUNT(e.employee\_id) AS Employee\_Count,

SUM(s.Sales) AS Total\_Sales

FROM regions r

LEFT JOIN employees e ON r.region\_id = e.region\_id

LEFT JOIN sales s ON e.employee\_id = s.Customer\_ID

GROUP BY r.region\_id, r.region, r.country;  
  
**4. Customers Table Analysis**-- 4.1 Segment customers based on age, location, and spending.  
  
 SELECT Age, Country, State, Region, Segment,

COUNT(\*) AS Customer\_Count,

AVG(Sales) AS Average\_Spending

FROM customers c

JOIN sales s ON c.Customer\_ID = s.Customer\_ID

GROUP BY Age, Country, State, Region, Segment;  
  
-- 4.2 Calculate the CLV for each customer segment.  
 WITH customer\_sales AS (

SELECT c.Customer\_ID, c.Segment, SUM(s.Sales) AS Total\_Spending

FROM customers c

JOIN sales s ON c.Customer\_ID = s.Customer\_ID

GROUP BY c.Customer\_ID, c.Segment

)

SELECT Segment, AVG(Total\_Spending) AS Avg\_CLV

FROM customer\_sales

GROUP BY Segment;

**5. Sales Table Analysis**

-- 5.1 Evaluate the performance of products across categories and subcategories.  
 SELECT Category, Sub\_Category, SUM(Quantity) AS Total\_Quantity, SUM(Sales) AS Total\_Sales, SUM(Profit) AS Total\_Profit FROM sales GROUP BY Category, Sub\_Category;  
  
-- 5.2 Identify sales trends across different seasons or months.

SELECT MONTH(Order\_Date) AS Month, Category, SUM(Sales) AS Total\_Sales

FROM sales

GROUP BY MONTH(Order\_Date), Category;  
  
-- 5.3 Sales Effectiveness by Discount Levels  
  
 SELECT Discount, AVG(Quantity) AS Avg\_Quantity, AVG(Sales) AS Avg\_Sales, AVG(Profit) AS Avg\_Profit

FROM sales

GROUP BY Discount;  
  
**SECTION 2: calculate relevant Key Performance Indicators (KPIs).  
Below are the important KPI’s queries that employ advanced SQL functions such as window functions, common table expressions (CTEs), and complex joins to analyze trends, performance, and correlations within your data.**

**1. Year-over-Year Sales Growth by Category:**

--Compare the annual growth in sales for each category.

WITH annual\_sales AS (

SELECT YEAR(Order\_Date) AS Year, Category, SUM(Sales) AS Total\_Sales

FROM sales

GROUP BY YEAR(Order\_Date), Category

)

SELECT current.Year, current.Category,

current.Total\_Sales,

previous.Total\_Sales AS Previous\_Year\_Sales,

(current.Total\_Sales - previous.Total\_Sales) / previous.Total\_Sales \* 100 AS Growth\_Percentage

FROM annual\_sales AS current

JOIN annual\_sales AS previous ON current.Category = previous.Category AND current.Year = previous.Year + 1;

**2. Employee Sales Performance with Ranking:**

--Rank employees based on the sales they have generated, partitioned by   
--department.

WITH employee\_sales AS (

SELECT e.employee\_id, e.first\_name, e.department, SUM(s.Sales) AS Total\_Sales

FROM employees e

JOIN sales s ON e.employee\_id = s.Customer\_ID

GROUP BY e.employee\_id, e.first\_name, e.department

)

SELECT employee\_id, first\_name, department, Total\_Sales,

RANK() OVER (PARTITION BY department ORDER BY Total\_Sales DESC) AS Sales\_Rank

FROM employee\_sales;

**3. Customer Lifetime Value (CLV):**

--Calculate the lifetime value of customers.

WITH customer\_lifetime\_sales AS (

SELECT Customer\_ID, SUM(Sales) AS Lifetime\_Sales, COUNT(DISTINCT Order\_ID) AS Total\_Orders

FROM sales

GROUP BY Customer\_ID)

SELECT c.Customer\_ID, c.Customer\_Name, cls.Lifetime\_Sales, cls.Total\_Orders,

cls.Lifetime\_Sales / cls.Total\_Orders AS Average\_Order\_Value

FROM customers c

JOIN customer\_lifetime\_sales cls ON c.Customer\_ID = cls.Customer\_ID;

**4. Sales Variance by Region and Category:**

--Analyze the variance in sales for each category across different regions.

SELECT c.Region, s.Category, AVG(s.Sales) AS Average\_Sales,

STDDEV(s.Sales) AS Sales\_Standard\_Deviation

FROM sales s

JOIN customers c ON s.Customer\_ID = c.Customer\_ID

GROUP BY c.Region, s.Category;

**5. Most Profitable Customers:**

--Identify the top 10 most profitable customers.

WITH customer\_profit AS (

SELECT Customer\_ID, SUM(Profit) AS Total\_Profit

FROM sales

GROUP BY Customer\_ID

)

SELECT c.Customer\_ID, c.Customer\_Name, cp.Total\_Profit

FROM customers c

JOIN customer\_profit cp ON c.Customer\_ID = cp.Customer\_ID

ORDER BY cp.Total\_Profit DESC

LIMIT 10;

**6. Inter-Departmental Salary Disparities:**

--Compare average salaries across departments.

SELECT department, AVG(salary) AS Average\_Salary,

MAX(salary) - MIN(salary) AS Salary\_Disparity

FROM employees

GROUP BY department;

**7. Monthly Sales and Profit Trends:**

--Track monthly sales and profit trends over the past year.

WITH monthly\_performance AS (

SELECT DATE\_FORMAT(Order\_Date, '%Y-%m') AS Month, SUM(Sales) AS Total\_Sales, SUM(Profit) AS Total\_Profit

FROM sales

WHERE Order\_Date >= DATE\_SUB(CURRENT\_DATE, INTERVAL 1 YEAR)

GROUP BY DATE\_FORMAT(Order\_Date, '%Y-%m')

)

SELECT Month, Total\_Sales, Total\_Profit,

LAG(Total\_Sales) OVER (ORDER BY Month) AS Previous\_Month\_Sales,

LAG(Total\_Profit) OVER (ORDER BY Month) AS Previous\_Month\_Profit

FROM monthly\_performance;

**SECTION 3: GENERAL DATA ANALYSIS USING ADVANCE SQL FUNCTIONS  
In this section, I have dived deep into the world of SQL. The Queries are divided into nine parts. I have gone beyond basic queries and cracked open different SQL windows to extract hidden insights from databases like a digital archaeologist:**

-- 1.1: Retrieve all the data in the projectdb database

SELECT \* FROM employees;

SELECT \* FROM departments;

SELECT \* FROM regions;

SELECT \* FROM customers;

SELECT \* FROM sales;

-- 2.1: Retrieve a list of employee\_id, first\_name, hire\_date,

-- and department of all employees ordered by the hire date

SELECT employee\_id, first\_name, department, hire\_date,

ROW\_NUMBER() OVER (ORDER BY hire\_date) AS Row\_N

FROM employees;

-- 2.2: Retrieve the employee\_id, first\_name,

-- hire\_date of employees for different departments

SELECT employee\_id, first\_name, , hire\_date FROM employees  
GROUP BY departments  
ORDER BY employee\_id;

-- 3.1: Recall the use of ROW\_NUMBER()

SELECT first\_name, email, department, salary,

ROW\_NUMBER() OVER(PARTITION BY department

ORDER BY salary DESC)

FROM employees;

-- 3.2: Retrieve the hire\_date. Return details of

-- employees hired on or before 31st Dec, 2005 and are in

-- First Aid, Movies and Computers departments

SELECT first\_name, email, department, salary, \_\_\_

RANK() OVER(PARTITION BY department

ORDER BY salary DESC)

FROM employees

WHERE \_\_\_ AND department \_\_\_;

-- This returns how many employees are in each department

SELECT department, COUNT(\*) dept\_count

FROM employees

GROUP BY department

ORDER BY dept\_count DESC;

-- 3.3: Return the fifth ranked salary for each department

-- Create a common table expression to retrieve the customer\_id,

-- and how many times the customer has purchased from the mall

WITH purchase\_count AS (

SELECT customer\_id, COUNT(sales) AS purchase

FROM sales

GROUP BY customer\_id

ORDER BY purchase DESC

)

-- 4.1: Group the employees table into five groups

-- based on the order of their salaries

SELECT first\_name, email, department, salary, hire\_date,

RANK() OVER(PARTITION BY department ORDER BY salary DESC) AS rank\_within\_department

FROM employees

WHERE hire\_date <= '2005-12-31'

AND department IN ('First Aid', 'Movies', 'Computers');

-- 4.2: Group the employees table into five groups for

-- each department based on the order of their salaries

SELECT first\_name, email, department, salary,

NTILE(5) OVER(PARTITION BY department

ORDER BY salary DESC)

FROM employees;

-- Create a CTE that returns details of an employee

-- and group the employees into five groups

-- based on the order of their salaries

WITH salary\_ranks AS (

SELECT first\_name, email, department, salary,

NTILE(5) OVER(ORDER BY salary DESC) AS rank\_of\_salary

FROM employees)

-- 4.3: Find the average salary for each group of employees

SELECT rank\_of\_salary,

AVG(salary) AS average\_salary

FROM salary\_ranks

GROUP BY rank\_of\_salary;

-- 5.1: This returns how many employees are in each department

SELECT department, COUNT(\*) AS dept\_count

FROM employees

GROUP BY department

ORDER BY department;

-- 5.2: Retrieve the first names, department and

-- number of employees working in that department

SELECT first\_name, department,

(SELECT COUNT(\*) AS dept\_count FROM employees e1 WHERE e1.department = e2.department)

FROM employees e2

GROUP BY department, first\_name

ORDER BY department;

-- 5.3: Total Salary for all employees

SELECT SUM(salary) AS total\_salary

FROM employees;

-- 5.4: Total Salary for each department

SELECT department,

SUM(salary) AS total\_department\_salary

FROM employees

GROUP BY department;  
  
-- Total Salary for each department and

-- order by the hire date.

SELECT first\_name, hire\_date, department, salary,

SUM(salary) OVER(PARTITION BY department ORDER BY hire\_date) AS running\_total

FROM employees;

-- Retrieve the different region ids

SELECT DISTINCT region\_id

FROM employees;

-- Retrieve the first names, department and

-- number of employees working in that department and in region 2

SELECT first\_name, department,

COUNT(\*) OVER(PARTITION BY department) AS dept\_count

FROM employees

WHERE region\_id = 2;

-- Create a common table expression to retrieve the customer\_id,

-- ship\_mode, and how many times the customer has purchased from the mall

WITH purchase\_count AS (

SELECT customer\_id, ship\_mode, COUNT(sales) AS purchase

FROM sales

GROUP BY customer\_id, ship\_mode

ORDER BY purchase DESC

)

-- Calculate the cumulative sum of customers purchase

SELECT first\_name, department,

COUNT(\*) OVER(PARTITION BY department) AS dept\_count

FROM employees

WHERE region\_id = 2;

-- 7.1: Calculate the running total of salary

SELECT first\_name, hire\_date, salary,

SUM(salary) OVER (ORDER BY hire\_date) AS running\_total\_salary

FROM employees

ORDER BY hire\_date;

-- 7.2: Add the current row and previous row

SELECT first\_name, hire\_date, salary,

LAG(salary) OVER (ORDER BY hire\_date) + salary AS total\_salary\_with\_previous

FROM employees

ORDER BY hire\_date;

-- 7.3: Find the running average

SELECT first\_name, hire\_date, salary,

AVG(salary) OVER (ORDER BY hire\_date) AS running\_average\_salary

FROM employees

ORDER BY hire\_date;

-- 8.1: Review of the FIRST\_VALUE() function

SELECT department, division,

FIRST\_VALUE(department) OVER(ORDER BY department ASC) first\_department

FROM departments;

-- Create a common table expression to retrieve the customer\_id,

-- ship\_mode, and how many times the customer has purchased from the mall

WITH purchase\_count AS (

SELECT customer\_id, COUNT(sales) AS purchase

FROM sales

GROUP BY customer\_id

ORDER BY purchase DESC

)

-- Calculate Max and Next Max Purchase for Each Customer using above CTE

SELECT customer\_id, purchase,

MAX(purchase) OVER(ORDER BY customer\_id ASC) AS max\_of\_sales,

MAX(purchase) OVER(ORDER BY customer\_id ASC

ROWS BETWEEN

CURRENT ROW AND 1 FOLLOWING) AS next\_max\_of\_sales

FROM purchase\_count;

-- 9.1: Find the sum of the quantity for different ship modes

SELECT ship\_mode, SUM(quantity)

FROM sales

GROUP BY ship\_mode;

-- 9.2: Find the sum of the quantity for different categories

SELECT category, SUM(quantity)

FROM sales

GROUP BY category;

-- 9.3: Find the sum of the quantity for different subcategories

SELECT sub\_category, SUM(quantity)

FROM sales

GROUP BY sub\_category;

-- 9.4: Use the GROUPING SETS clause

SELECT ship\_mode, category, sub\_category, SUM(quantity)

FROM sales

GROUP BY GROUPING SETS (ship\_mode, category, sub\_category);

--9.5: Use the ROLLUP clause

SELECT ship\_mode, category, sub\_category, SUM(quantity)

FROM sales

GROUP BY ROLLUP (ship\_mode, category, sub\_category);

--9.6: Use the CUBE clause

SELECT ship\_mode, category, sub\_category, SUM(quantity)

FROM sales

GROUP BY CUBE (ship\_mode, category, sub\_category);