Advance: MySQL Subqueries, HAVING Clause and INNER JOIN – Practice Lab

Objective:
This lab reinforces your understanding of subqueries, aggregate filtering (HAVING), and INNER JOIN queries. You will repeat the exercises with a focus on mastering syntax, query logic, and correct outputs.

Instructions:
- Execute each query using the classicmodels database.
- Take screenshots of your SQL output and label them clearly (e.g., Lab5-Q1, Lab5-Q2, Lab5-Q3……).
- Compile your screenshots into a single PDF or Word document.
- Submit the compiled document.

Part A: Subqueries

- Q1. List all employees working in offices located in the USA.

```
SELECT lastName, firstName
FROM employees
WHERE officeCode IN (
   SELECT officeCode
   FROM offices
   WHERE country = 'USA'
);
```

- Q2. Find the customer who made the maximum payment.

```
SELECT customerNumber, checkNumber, amount
FROM payments
WHERE amount = (
   SELECT MAX(amount)
   FROM payments
);
```

- Q3. Find customers who paid more than the average amount.

```
SELECT customerNumber, checkNumber, amount
FROM payments
WHERE amount > (
   SELECT AVG(amount)
   FROM payments
);
```

- Q4. List all customers who have never ordered any product.

```
SELECT customerName
FROM customers
WHERE customerNumber NOT IN (
  SELECT DISTINCT customerNumber
  FROM orders
);
```

Part B: EXISTS and NOT EXISTS

- Q5. List all customers who have at least one order with sales > $10,000.

```
SELECT customerName
FROM customers
WHERE EXISTS (
  SELECT priceEach * quantityOrdered
  FROM orderdetails
  WHERE priceEach * quantityOrdered > 10000
  GROUP BY orderNumber
);
```

- Q6. Try running the same query with NOT EXISTS. What happens? Why?

Part C: Subquery in FROM Clause (Derived Table)

- Q7. Find max, min, and average number of items per order.

```
SELECT MAX(items), MIN(items), FLOOR(AVG(items))
FROM (
  SELECT orderNumber, COUNT(orderNumber) AS items
  FROM orderdetails
  GROUP BY orderNumber
) AS lineitems;
```

Part D: Correlated Subquery

- Q8. List products with a buy price higher than the average for their product line.

```
SELECT productName, buyPrice
FROM products AS p1
WHERE buyPrice > (
  SELECT AVG(buyPrice)
```

```
    FROM products
    WHERE productLine = p1.productLine
);
```

Part E: HAVING Clause

- Q9. Show order numbers with total sales greater than $1000.

```
SELECT orderNumber,
    SUM(priceEach) AS total
FROM orderdetails
GROUP BY orderNumber
HAVING total > 1000;
```

- Q10. Show orders with total sales > $1000 and more than 600 items.

```
SELECT orderNumber,
    SUM(quantityOrdered) AS itemsCount,
    SUM(priceEach) AS total
FROM orderdetails
GROUP BY orderNumber
HAVING total > 1000 AND itemsCount > 600;
```

Part F: INNER JOIN and GROUP BY

- Q11. Show shipped orders with total sales > $1500.

```
SELECT a.orderNumber, SUM(priceEach) AS total, b.status
FROM orderdetails a
INNER JOIN orders b ON a.orderNumber = b.orderNumber
GROUP BY a.orderNumber
HAVING b.status = 'Shipped' AND total > 1500;
```

- Q12. Show product name, code, and product line description.

```
SELECT productCode, productName, textDescription
FROM products T1
INNER JOIN productlines T2 ON T1.productLine = T2.productLine;
```