Subqueries

1. **Average Freight for Specific Employee:**

```
SELECT employee_id, AVG(freight)::numeric(30,3)
FROM orders
WHERE employee_id = 3
GROUP BY employee_id;
```

   - `AVG`: An aggregate function that calculates the average value.
   - `::numeric`: Casts the average to a numeric type with specified precision.
   - Fetches `employee_id` and the average freight for employee with ID 3.

2. **Employees with Average Freight Greater than 85.707:**

```
SELECT CONCAT(e.first_name, ' ', e.last_name) as full_name
FROM orders AS o JOIN employees AS e ON e.employee_id = o.employee_id
GROUP BY e.first_name, e.last_name
HAVING AVG(o.freight) > 85.707;
```

   - `CONCAT`: Concatenates strings.
   - `HAVING`: Filters groups based on aggregate functions.
   - Fetches full names of employees with average freight greater than 85.707.

3. **Employees with Above Average Freight Compared to a Specific Employee:**

```
SELECT CONCAT(e.first_name, ' ', e.last_name) as full_name
FROM orders AS o JOIN employees AS e ON e.employee_id = o.employee_id
GROUP BY e.first_name, e.last_name
HAVING AVG(o.freight) >
(
  SELECT AVG(freight)::numeric(30,3)
  FROM orders
  WHERE employee_id = 3
);
```

- Fetches full names of employees with average freight greater than the average freight of employee with ID 3.

4. **Products from a Specific Supplier:**

```
SELECT product_name
FROM products
WHERE supplier_id =
(
  SELECT supplier_id
  FROM suppliers
  WHERE company_name LIKE 'Exotic Liquids'
);
```

- Fetches product names from the supplier 'Exotic Liquids'.

5. **Employees Handling Orders with Maximum Freight:**

```
SELECT first_name, last_name
FROM employees AS e, orders AS o
WHERE e.employee_id = o.employee_id
AND o.freight =
(
  SELECT MAX(freight) FROM orders
);
```

- `MAX`: An aggregate function that returns the maximum value.
- Fetches first and last names of employees who handled orders with the maximum freight.

6. **Employees with Above Average Freight:**

```
SELECT e.first_name, e.last_name, AVG(o.freight)
FROM orders AS o
JOIN employees AS e
ON e.employee_id = o.employee_id
GROUP BY e.first_name, e.last_name
HAVING AVG(o.freight) >
(
  SELECT AVG(freight) FROM orders
);
```

- Fetches first and last names of employees with average freight greater than the overall average freight.

7. **Most Expensive Product:**

```sql
SELECT product_name
FROM products
WHERE unit_price =
(
  SELECT MAX(unit_price) FROM products
);
```

- Fetches the name of the most expensive product.

8. **Count of Dessert Products:**

```sql
SELECT COUNT(product_id) AS count_desserts
FROM products
GROUP BY category_id
HAVING category_id =
(
  SELECT category_id
  FROM categories
  WHERE description LIKE 'Desserts%'
);
```

- Fetches the count of dessert products.

9. **Categories with Above Average Price:**

```sql
SELECT c.category_name
FROM products AS p
JOIN categories AS c
ON p.category_id = c.category_id
GROUP BY c.category_name
HAVING AVG(p.unit_price) >
(
  SELECT AVG(unit_price) FROM products
);
```

- Fetches category names with average product prices greater than the overall average price.

10. **Suppliers with Below Average Freight:**

```sql
SELECT s.company_name
FROM orders AS o
JOIN order_details AS od ON o.order_id = od.order_id
JOIN products AS p ON p.product_id = od.product_id
JOIN suppliers AS s ON s.supplier_id = p.supplier_id
GROUP BY s.company_name
HAVING AVG(o.freight) <
(
  SELECT AVG(freight) FROM orders
);
```

   - Fetches company names of suppliers with average freight charges less than the overall average.

11. **Shippers with Phone Number Containing '99':**

```sql
SELECT shipper_id
FROM shippers
WHERE phone LIKE '%99%';
```

   - Fetches shipper IDs with phone numbers containing '99'.

12. **Customers Using Specific Shippers:**

```sql
SELECT DISTINCT company_name, city
FROM customers AS c
JOIN orders AS o ON c.customer_id = o.customer_id
WHERE o.ship_via IN
(
  SELECT shipper_id
  FROM shippers
  WHERE phone LIKE '%99%'
);
```

   - `DISTINCT`: Removes duplicate rows from the result set.
   - Fetches distinct company names and cities of customers using shippers with phone numbers containing '99'.

13. **Suppliers Not in Customer Cities:**

```
SELECT DISTINCT company_name, city
FROM suppliers
WHERE city NOT IN
(
  SELECT city
  FROM customers
  WHERE city IS NOT NULL
);
```

- Fetches distinct company names and cities of suppliers located in cities where there are no customers.

14. **Suppliers in No Cities:**

```
SELECT DISTINCT company_name, city
FROM suppliers
WHERE city NOT IN
(
  SELECT NULL
);
```

- Fetches distinct company names and cities of suppliers located in cities that do not match NULL (always false condition).

15. **Average Freight for Employees Outside a Specific Region:**

```
SELECT AVG(freight)
FROM orders
WHERE employee_id NOT IN
(
  SELECT employee_id
  FROM employees
  WHERE region = 'WA'
);
```

- Fetches the average freight for orders handled by employees not in the 'WA' region.

16. **Employees Handling Orders to the USA:**

```
SELECT first_name, last_name
FROM employees
WHERE employee_id IN
```

```
(
  SELECT DISTINCT employee_id
  FROM orders
  WHERE ship_country = 'USA'
);
```

- Fetches first and last names of employees who handled orders shipped to the USA.

17. **Employees with Orders from Specific Customer:**

```
SELECT DISTINCT last_name || ' ' || first_name AS full_name, region
FROM employees
WHERE employee_id = ANY
(
  SELECT employee_id
  FROM orders
  WHERE customer_id LIKE 'VINET'
);
```

- `ANY`: Compares a value to any value in a list or subquery.
- Fetches full names and regions of employees who handled orders from the customer with ID 'VINET'.

18. **Employees Handling Orders to the USA:**

```
SELECT first_name, last_name
FROM employees
WHERE employee_id = ANY
(
  SELECT DISTINCT employee_id
  FROM orders
  WHERE ship_country = 'USA'
);
```

- Fetches first and last names of employees who handled orders shipped to the USA.

19. **Orders with Minimum Priced Products from USA Suppliers:**

```
SELECT o.customer_id, o.order_date
FROM orders AS o
JOIN order_details AS od ON o.order_id = od.order_id
JOIN products AS p ON p.product_id = od.product_id
JOIN suppliers AS s ON p.supplier_id = s.supplier_id
WHERE p.unit_price =
(
  SELECT MIN(pr.unit_price)
  FROM suppliers AS sup
  JOIN products AS pr ON sup.supplier_id = pr.supplier_id
  WHERE sup.country = 'USA'
)
AND s.country = 'USA';
```

- `MIN`: An aggregate function that returns the minimum value.
- Fetches customer IDs and order dates for orders containing the minimum priced products from USA suppliers.

20. **Orders Grouped by Employee with Latest Order Date:**

```
SELECT order_id, customer_id, employee_id, order_date, required_date
FROM orders
WHERE order_date IN
(
  SELECT MAX(order_date)
  FROM orders
  GROUP BY employee_id
);
```

- Fetches order details for the latest order date grouped by employee.

21. **Correlated Subquery for Latest Order Date by Employee:**

```
SELECT order_id, customer_id, employee_id, order_date, required_date
FROM orders AS o1
WHERE order_date =
(
  SELECT MAX(order_date)
  FROM orders AS o2
  WHERE o2.employee_id = o1.employee_id
);
```

- `Correlated Subquery`: A subquery that uses values from the outer query.
- Fetches order details for the latest order date for each employee.

22. **Orders with Quantity Less than 10% of Average:**

```sql
SELECT DISTINCT order_id
FROM order_details AS od
WHERE quantity <
(
  SELECT AVG(quantity) * 0.1
  FROM order_details
  WHERE od.product_id = product_id
);
```

- Fetches distinct order IDs with quantities less than 10% of the average quantity for the same product.

23. **Top 2 Customers by Region with Highest Order Value:**

```sql
SELECT company_name, contact_name, address
FROM customers AS outer_c
WHERE outer_c.customer_id IN
(
  SELECT inner_c.customer_id
  FROM order_details AS od
  JOIN orders AS o ON od.order_id = o.order_id
  JOIN customers AS inner_c ON o.customer_id = inner_c.customer_id
  WHERE inner_c.region = outer_c.region
  GROUP BY inner_c.region, inner_c.customer_id
  ORDER BY SUM(od.unit_price * od.quantity * (1-od.discount)) DESC
  LIMIT 2
)
ORDER BY outer_c.region;
```

- Fetches company names, contact names, and addresses of the top 2 customers by region with the highest order value.

24. **Nested Subquery for Orders with a Specific Product:**

```sql
SELECT * FROM orders WHERE order_id IN
(
  SELECT order_id FROM order_details WHERE product_id =
  (
    SELECT product_id FROM products WHERE product_name='Chai'
  )
);
```

  - `Nested Subquery`: A subquery within another subquery.
  - Fetches order details for orders containing the product named 'Chai'.