**Sample SQL queries:**

SELECT \* FROM transactions ORDER BY transaction\_date DESC;

SELECT product\_id, product\_category FROM products WHERE unit\_price > 5;

SELECT store\_id, store\_location FROM stores;

SELECT transaction\_id, transaction\_qty FROM transactions WHERE transaction\_qty > 5;

SELECT p.product\_id, p.product\_category, SUM(t.transaction\_qty \* p.unit\_price) AS total\_revenue FROM transactions t JOIN products p ON t.product\_id = p.product\_id GROUP BY p.product\_id, p.product\_category;

SELECT DISTINCT s.store\_id, s.store\_location FROM transactions t JOIN stores s ON t.store\_id = s.store\_id WHERE t.product\_id = 87;

SELECT s.store\_id, s.store\_location, SUM(t.transaction\_qty) AS total\_quantity\_sold FROM transactions t JOIN stores s ON t.store\_id = s.store\_id GROUP BY s.store\_id, s.store\_location;

SELECT p.product\_id, p.product\_category, COUNT(t.transaction\_id) AS transaction\_count FROM products p LEFT JOIN transactions t ON p.product\_id = t.product\_id GROUP BY p.product\_id, p.product\_category;

SELECT SUM(transaction\_qty) AS total\_quantity\_sold FROM transactions;

SELECT MIN(unit\_price) AS min\_price, MAX(unit\_price) AS max\_price FROM products;

SELECT AVG(transaction\_qty) AS avg\_quantity\_sold FROM transactions;

SELECT COUNT(store\_id) AS total\_stores FROM stores;

SELECT SUM(t.transaction\_qty \* p.unit\_price) AS total\_revenue FROM transactions t JOIN products p ON t.product\_id = p.product\_id;

SELECT store\_id, COUNT(transaction\_id) AS total\_transactions FROM transactions GROUP BY store\_id HAVING COUNT(transaction\_id) > 500;

SELECT product\_category, AVG(unit\_price) AS avg\_price FROM products GROUP BY product\_category HAVING AVG(unit\_price) > 5;

SELECT product\_id, SUM(transaction\_qty) AS total\_sold FROM transactions GROUP BY product\_id HAVING SUM(transaction\_qty) > 1000;

SELECT s.store\_location, SUM(t.transaction\_qty \* p.unit\_price) AS total\_revenue FROM transactions t JOIN stores s ON t.store\_id = s.store\_id JOIN products p ON t.product\_id = p.product\_id GROUP BY s.store\_location HAVING SUM(t.transaction\_qty \* p.unit\_price) > 10000;

SELECT product\_type, COUNT(DISTINCT product\_id) AS num\_products FROM products GROUP BY product\_type HAVING COUNT(DISTINCT product\_id) > 5;

SELECT store\_id, COUNT(transaction\_id) AS total\_transactions FROM transactions GROUP BY store\_id;

SELECT store\_id, COUNT(DISTINCT product\_id) AS distinct\_products\_sold FROM transactions GROUP BY store\_id;

SELECT p.product\_category, AVG(t.transaction\_qty) AS avg\_quantity\_sold FROM transactions t JOIN products p ON t.product\_id = p.product\_id GROUP BY p.product\_category;

SELECT t.store\_id, SUM(t.transaction\_qty \* p.unit\_price) AS store\_revenue FROM transactions t JOIN products p ON t.product\_id = p.product\_id GROUP BY t.store\_id;

SELECT \* FROM transactions ORDER BY transaction\_date ASC;

SELECT product\_id, unit\_price, product\_category FROM products ORDER BY unit\_price DESC;

SELECT store\_id, store\_location FROM stores ORDER BY store\_location ASC;

SELECT transaction\_id, store\_id, transaction\_qty FROM transactions ORDER BY store\_id ASC, transaction\_qty DESC;

SELECT product\_id, product\_category, product\_type FROM products ORDER BY product\_category ASC, product\_type ASC;

**Sample NL queries:**

Get all transactions and order them by transaction date in descending order.

Get the product ID and category for products with a unit price greater than 5.

Get the store ID and location for all stores.

Get the transaction ID and quantity for transactions where the quantity is greater than 5.

Get the product ID, category, and total revenue (calculated as quantity \* unit price) for each product, grouped by product ID and category.

Find the distinct store IDs and locations where product ID 87 is sold.

Get the store ID, location, and the total quantity sold at each store, grouped by store ID and location.

Get the product ID, category, and the count of transactions for each product, even if no transactions exist for a product.

Get the total quantity sold across all transactions.

Find the minimum and maximum unit price of all products.

Find the average quantity sold across all transactions.

Get the total number of stores.

Get the total revenue from all transactions (quantity \* unit price).

Find the store ID and the number of transactions for stores with more than 500 transactions.

Find the average price for each product category, but only for categories with an average price greater than 5.

Get the product ID and total quantity sold for products where the total quantity sold exceeds 1000.

Find the store location and total revenue for stores where the total revenue exceeds 10,000.

Get the product type and the number of distinct products for product types with more than 5 products.

Get the store ID and the total number of transactions for each store.

Get the store ID and the count of distinct products sold at each store.

Get the product category and the average quantity sold for each product category.

Get the store ID and the total revenue from transactions at each store.

Get all transactions and order them by transaction date in ascending order.

Get the product ID, unit price, and category for all products, ordered by unit price in descending order.

Get the store ID and location for all stores, ordered by store location in ascending order.

Get the transaction ID, store ID, and quantity, ordered by store ID in ascending order and transaction quantity in descending order.

Get the product ID, category, and type for all products, ordered by product category and type in ascending order.

**Sample SQL+NL queries:**

**SQL Query**:  
SELECT \* FROM transactions ORDER BY transaction\_date DESC;  
**English**: Get all transactions and order them by transaction date in descending order.

**SQL Query**:  
SELECT product\_id, product\_category FROM products WHERE unit\_price > 5;  
**English**: Get the product ID and category for products with a unit price greater than 5.

**SQL Query**:  
SELECT store\_id, store\_location FROM stores;  
**English**: Get the store ID and location for all stores.

**SQL Query**:  
SELECT transaction\_id, transaction\_qty FROM transactions WHERE transaction\_qty > 5;  
**English**: Get the transaction ID and quantity for transactions where the quantity is greater than 5.

**SQL Query**:  
SELECT p.product\_id, p.product\_category, SUM(t.transaction\_qty \* p.unit\_price) AS total\_revenue FROM transactions t JOIN products p ON t.product\_id = p.product\_id GROUP BY p.product\_id, p.product\_category;  
**English**: Get the product ID, category, and total revenue (calculated as quantity \* unit price) for each product, grouped by product ID and category.

**SQL Query**:  
SELECT DISTINCT s.store\_id, s.store\_location FROM transactions t JOIN stores s ON t.store\_id = s.store\_id WHERE t.product\_id = 87;  
**English**: Find the distinct store IDs and locations where product ID 87 is sold.

**SQL Query**:  
SELECT s.store\_id, s.store\_location, SUM(t.transaction\_qty) AS total\_quantity\_sold FROM transactions t JOIN stores s ON t.store\_id = s.store\_id GROUP BY s.store\_id, s.store\_location;  
**English**: Get the store ID, location, and the total quantity sold at each store, grouped by store ID and location.

**SQL Query**:  
SELECT p.product\_id, p.product\_category, COUNT(t.transaction\_id) AS transaction\_count FROM products p LEFT JOIN transactions t ON p.product\_id = t.product\_id GROUP BY p.product\_id, p.product\_category;  
**English**: Get the product ID, category, and the count of transactions for each product, even if no transactions exist for a product.

**SQL Query**:  
SELECT SUM(transaction\_qty) AS total\_quantity\_sold FROM transactions;  
**English**: Get the total quantity sold across all transactions.

**SQL Query**:  
SELECT MIN(unit\_price) AS min\_price, MAX(unit\_price) AS max\_price FROM products;  
**English**: Find the minimum and maximum unit price of all products.

**SQL Query**:  
SELECT AVG(transaction\_qty) AS avg\_quantity\_sold FROM transactions;  
**English**: Find the average quantity sold across all transactions.

**SQL Query**:  
SELECT COUNT(store\_id) AS total\_stores FROM stores;  
**English**: Get the total number of stores.

**SQL Query**:  
SELECT SUM(t.transaction\_qty \* p.unit\_price) AS total\_revenue FROM transactions t JOIN products p ON t.product\_id = p.product\_id;  
**English**: Get the total revenue from all transactions (quantity \* unit price).

**SQL Query**:  
SELECT store\_id, COUNT(transaction\_id) AS total\_transactions FROM transactions GROUP BY store\_id HAVING COUNT(transaction\_id) > 500;  
**English**: Find the store ID and the number of transactions for stores with more than 500 transactions.

**SQL Query**:  
SELECT product\_category, AVG(unit\_price) AS avg\_price FROM products GROUP BY product\_category HAVING AVG(unit\_price) > 5;  
**English**: Find the average price for each product category, but only for categories with an average price greater than 5.

**SQL Query**:  
SELECT product\_id, SUM(transaction\_qty) AS total\_sold FROM transactions GROUP BY product\_id HAVING SUM(transaction\_qty) > 1000;  
**English**: Get the product ID and total quantity sold for products where the total quantity sold exceeds 1000.

**SQL Query**:  
SELECT s.store\_location, SUM(t.transaction\_qty \* p.unit\_price) AS total\_revenue FROM transactions t JOIN stores s ON t.store\_id = s.store\_id JOIN products p ON t.product\_id = p.product\_id GROUP BY s.store\_location HAVING SUM(t.transaction\_qty \* p.unit\_price) > 10000;  
**English**: Find the store location and total revenue for stores where the total revenue exceeds 10,000.

**SQL Query**:  
SELECT product\_type, COUNT(DISTINCT product\_id) AS num\_products FROM products GROUP BY product\_type HAVING COUNT(DISTINCT product\_id) > 5;  
**English**: Get the product type and the number of distinct products for product types with more than 5 products.

**SQL Query**:  
SELECT store\_id, COUNT(transaction\_id) AS total\_transactions FROM transactions GROUP BY store\_id;  
**English**: Get the store ID and the total number of transactions for each store.

**SQL Query**:  
SELECT store\_id, COUNT(DISTINCT product\_id) AS distinct\_products\_sold FROM transactions GROUP BY store\_id;  
**English**: Get the store ID and the count of distinct products sold at each store.

**SQL Query**:  
SELECT p.product\_category, AVG(t.transaction\_qty) AS avg\_quantity\_sold FROM transactions t JOIN products p ON t.product\_id = p.product\_id GROUP BY p.product\_category;  
**English**: Get the product category and the average quantity sold for each product category.

**SQL Query**:  
SELECT t.store\_id, SUM(t.transaction\_qty \* p.unit\_price) AS store\_revenue FROM transactions t JOIN products p ON t.product\_id = p.product\_id GROUP BY t.store\_id;  
**English**: Get the store ID and the total revenue from transactions at each store.

**SQL Query**:  
SELECT \* FROM transactions ORDER BY transaction\_date ASC;  
**English**: Get all transactions and order them by transaction date in ascending order.

**SQL Query**:  
SELECT product\_id, unit\_price, product\_category FROM products ORDER BY unit\_price DESC;  
**English**: Get the product ID, unit price, and category for all products, ordered by unit price in descending order.

**SQL Query**:  
SELECT store\_id, store\_location FROM stores ORDER BY store\_location ASC;  
**English**: Get the store ID and location for all stores, ordered by store location in ascending order.

**SQL Query**:  
SELECT transaction\_id, store\_id, transaction\_qty FROM transactions ORDER BY store\_id ASC, transaction\_qty DESC;  
**English**: Get the transaction ID, store ID, and quantity, ordered by store ID in ascending order and transaction quantity in descending order.

**SQL Query**:  
SELECT product\_id, product\_category, product\_type FROM products ORDER BY product\_category ASC, product\_type ASC;  
**English**: Get the product ID, category, and type for all products, ordered by product category and type in ascending order.