20 Questions to solve

Focus: SELECT, WHERE, ORDER BY, GROUP BY, LIMIT, simple calculations

1. **Get the first 10 rows of Tesla stock data.**  
   → Use SELECT and LIMIT.
2. **List all distinct stock names in the dataset.**  
   → Use DISTINCT.
3. **Find the date and closing price when Tesla had the highest price.**  
   → Use MAX() with WHERE.
4. **What is the average closing price of NVIDIA in 2023?**  
   → Use AVG() with WHERE and YEAR().
5. **Count how many trading days are available for each stock.**  
   → Use GROUP BY and COUNT().
6. **What is the highest and lowest volume traded for each stock?**  
   → Use GROUP BY with MAX() and MIN().

**🟡 Intermediate-Level (Data Patterns & Comparisons)**

Focus: GROUP BY, DATE\_FORMAT, CASE, calculated fields, aliases

1. **Calculate the daily return (%) for Tesla.**  
   → Formula: (close\_price - open\_price) / open\_price \* 100

SELECT `Date`, `open`, `close`,

ROUND(((`close` - `open`)/ `open`) \* 100, 2) AS Daily\_Return\_Pct

FROM tesla\_database

WHERE `open` IS NOT NULL AND `close` IS NOT NULL

ORDER BY `Date`;

A screenshot of a computer

AI-generated content may be incorrect.

1. **List the top 5 most volatile days by daily % change (abs).**  
   → Use ABS() and ORDER BY.

SELECT `Date`, `open`, `close`,

ROUND(((`close` - `open`)/ `open`) \* 100, 2) AS Daily\_Return\_Pct,

ROUND(ABS((`close` - `open`)/ `open`) \* 100, 2) AS daily\_volatility\_pct \*\*ABS (Absolute ignores positive and negative sign)

FROM tesla\_database

WHERE `open` IS NOT NULL AND `close` IS NOT NULL

ORDER BY daily\_volatility\_pct DESC

A screenshot of a computer

AI-generated content may be incorrect.LIMIT 5;

1. **Find the average monthly closing price for both stocks.**  
   → Use GROUP BY DATE\_FORMAT(date, '%Y-%m').

SELECT

DATE\_FORMAT(`Date`, '%Y-%m') AS `Month`,

ROUND(AVG(`close`), 2) AS Avg\_Monthly\_Close

FROM tesla\_database

WHERE `close` IS NOT NULL

GROUP BY DATE\_FORMAT(`Date`, '%Y-%m')

ORDER BY `Month`;

A screenshot of a data

AI-generated content may be incorrect.

1. **Find the number of days each stock closed higher than it opened.**  
   → Use CASE WHEN close\_price > open\_price THEN 1 ELSE 0 END.

SELECT

SUM(CASE WHEN close > open THEN 1 ELSE 0 END) AS days\_closed\_higher

FROM tesla\_database;

-- OR

SELECT

COUNT(\*) AS days\_closed\_higher

FROM tesla\_database

WHERE close > open;

A screenshot of a computer

AI-generated content may be incorrect.

1. **Compare average daily volumes of Tesla in 2024.**  
   → Use WHERE with GROUP BY.

SELECT

AVG(volume) AS avg\_daiy\_volume\_2024

FROM tesla\_database

WHERE YEAR(date) = 2024;

A screenshot of a computer

AI-generated content may be incorrect.

-- Monthly Comparison in 2024

SELECT

MONTH(DATE) AS month,

AVG(volume) AS avg\_daily\_volume

FROM tesla\_database

WHERE YEAR(date) = 2024

GROUP BY MONTH(date)

ORDER BY MONTH(Date);

A screenshot of a computer

AI-generated content may be incorrect.

**🔵 Advanced-Level (Analytics, Trends & Business Insights)**

Focus: WINDOW FUNCTIONS, CTEs, LAG(), LEAD(), JOIN, subqueries

1. **Calculate the 7-day moving average of closing price for each stock.**  
   → Use AVG() OVER(PARTITION BY stock\_name ORDER BY date ROWS 6 PRECEDING).

SELECT

`Date`,

`Close`,

ROUND(AVG(`Close`) OVER (

ORDER BY `Date`

ROWS BETWEEN 6 PRECEDING AND CURRENT ROW), 4) AS moving\_avg\_7\_day

FROM tesla\_database;

A screenshot of a computer screen

AI-generated content may be incorrect.

We are selecting 3 things:

1. The Date of the record.
2. The Close price on that date.
3. The calculated 7-day moving average.

**AVG(Close) OVER (...)**

* This tells MySQL: "Use the **average of the Close prices** in a **sliding window** over the data."

**ORDER BY Date**

* This makes sure the rows are processed in the **correct date order**, from oldest to newest.
* Important: the moving average must follow chronological order!

**ROWS BETWEEN 6 PRECEDING AND CURRENT ROW**

This is the core of the "7-day" part:

* For **each row**, MySQL looks at:
  + That row (CURRENT ROW)
  + And the **6 previous rows** (6 PRECEDING)
* So, that makes **7 rows total**.
* Then it averages the Close prices of those 7 days.

**ROUND(..., 4)**

* This just **rounds** the final result to 4 decimal places.
* You can change 4 to any number (like 2 for cents).

1. **Find days where trading volume was twice the monthly average for that stock.**  
   → Use CTE or JOIN with subquery on monthly average.

WITH monthyly\_avg AS (

SELECT

YEAR(Date) AS year,

MONTH(Date) AS month,

ROUND(AVG(Volume), 2) AS avg\_volume

FROM tesla\_database

GROUP BY YEAR(Date), MONTH(Date)

)

SELECT

sp.Date,

sp.Volume,

ma.avg\_volume,

ROUND(sp.Volume / ma.avg\_volume, 2) AS ratio

FROM tesla\_database sp

JOIN monthyly\_avg ma

ON YEAR(sp.Date) = ma.year

AND MONTH(sp.Date) = ma.month

WHERE sp.Volume > 2 \* ma.avg\_volume;

1. **Find the maximum drawdown for each stock.**  
   → Use LAG() and a calculated drawdown column.

WITH price\_lagged AS (

SELECT

Date,

Close,

LAG(Close) OVER (ORDER BY Date) AS prev\_close

FROM tesla\_database

),

peaks\_and\_drawdown AS (

SELECT

Date,

Close,

MAX(Close) OVER (ORDER BY Date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS peak\_price,

(Close - MAX(Close) OVER (ORDER BY Date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW))

/ MAX(Close) OVER (ORDER BY Date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS drawdown

FROM price\_lagged

)

SELECT

MIN(drawdown) AS max\_drawdown

FROM peaks\_and\_drawdown;

SELECT

Date,

drawdown

FROM (

SELECT

Date,

Close,

MAX(Close) OVER (ORDER BY Date) AS peak\_price,

(Close - MAX(Close) OVER (ORDER BY Date)) / MAX(Close) OVER (ORDER BY Date) AS drawdown

FROM tesla\_database

) AS sub

ORDER BY drawdown ASC

LIMIT 1;

A screenshot of a computer

AI-generated content may be incorrect.

This means Tesla dropped by **73.63%** from its peak at some point.

1. **Identify days with a price gap-up or gap-down > 3% from previous close.**  
   → Use LAG() and percent change.

WITH price\_gaps AS (

SELECT

Date,

`Open`,

`Close`,

LAG(`Close`) OVER (ORDER BY Date) AS prev\_close,

ROUND(((`Open` - LAG(`Close`) OVER (ORDER BY Date)) / LAG(`Close`) OVER (ORDER BY Date)) \* 100, 2) AS gap\_percent

FROM tesla\_database

)

SELECT \*

FROM price\_gaps

WHERE ABS(gap\_percent) > 3;

**Concept:**

A **gap** is calculated as:

Gap %=Today’s Open−Yesterday’s CloseYesterday’s Close×100\text{Gap \%} = \frac{\text{Today's Open} - \text{Yesterday's Close}}{\text{Yesterday's Close}} \times 100Gap %=Yesterday’s CloseToday’s Open−Yesterday’s Close​×100

If gap% > 3% → **Gap-Up**  
If gap% < -3% → **Gap-Down**

1. **Rank Tesla and NVIDIA daily based on performance.**  
   → Use RANK() OVER(PARTITION BY date ORDER BY daily\_return DESC).

SELECT

Date,

`Open`,

`Close`,

ROUND((`Close` - `Open`) / `Open` \* 100, 2) AS daily\_return\_percent,

RANK() OVER (ORDER BY (`Close` - `Open`) / `Open` DESC) AS performance\_rank

FROM tesla\_database;

A common way to measure daily performance:

Performance % = (Close − Open / Open) × 100

Then, we rank days based on this value — highest % gain gets rank 1.

1. **Compare cumulative return for stock over a selected year.**  
   → Use SUM(daily\_return) OVER(PARTITION BY stock\_name ORDER BY date).

WITH daily\_returns AS (

SELECT

Date,

`Open`,

`Close`,

ROUND((`Close` - `Open`) / `Open` \* 100, 4) AS daily\_return

FROM tesla\_database

WHERE YEAR(Date) = 2023

)

SELECT

Date,

daily\_return,

ROUND(SUM(daily\_return) OVER (ORDER BY Date), 4) AS cumulative\_return

FROM daily\_returns

ORDER BY Date;

1. **Flag all non-trading days (weekends/holidays) missing in the dataset.**  
   → Use a calendar table and LEFT JOIN.
2. **If you invested $1000 at the beginning of each month, how much would it be worth now?**  
   → Use monthly start price + cumulative return logic.

WITH first\_trading\_days AS (

SELECT MIN(Date) AS Invest\_date

FROM tesla\_database

GROUP BY YEAR(Date), MONTH(Date)

),

**SOLUTION**

SELECT \*

FROM tesla\_stock\_data\_2000\_2025;

ALTER TABLE tesla\_stock\_data\_2000\_2025

RENAME COLUMN Price TO `Date`;

DELETE FROM tesla\_stock\_data\_2000\_2025

WHERE (`Date` = 'Ticker' AND `Close` = 'TSLA');

DELETE FROM tesla\_stock\_data\_2000\_2025

WHERE (`Date` = `Date` AND `Close` = '');

CREATE TABLE Tesla\_Database LIKE tesla\_stock\_data\_2000\_2025;

INSERT INTO Tesla\_Database SELECT \* FROM tesla\_stock\_data\_2000\_2025;

SELECT \* FROM Tesla\_Database;

DROP TABLE tesla\_databse;

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY `Date`, `Close`, High, Low, `Open`, Volume) AS row\_num

FROM tesla\_database;

WITH duplicate\_cte AS

(

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY `Date`, `Close`, High, Low, `Open`, Volume) AS row\_num

FROM tesla\_database

)

SELECT \*

FROM duplicate\_cte

WHERE row\_num > 1;

-- Standardizing Data

SELECT

TRIM(`Date`) AS Date,

TRIM(`Close`) AS Close,

TRIM(High) AS High,

TRIM(Low) AS Low,

TRIM(`Open`) AS Open,

TRIM(Volume) AS Volume

FROM tesla\_database;

SELECT `date`,

STR\_TO\_DATE(`date`, '%Y-%m-%d')

FROM tesla\_database;

UPDATE tesla\_database

SET `date` = STR\_TO\_DATE(`date`, '%Y-%m-%d');

SELECT `date`

FROM tesla\_database;

ALTER TABLE tesla\_database

MODIFY COLUMN `Date` DATE;

-- Change the number of digit after decimal

ALTER TABLE tesla\_database

MODIFY `Close` DECIMAL(17,4),

MODIFY High DECIMAL(17,4),

MODIFY low DECIMAL(17,4),

MODIFY `open` DECIMAL(17,4);

-- Questions

-- 1. Get the first 10 rows of Tesla stock data.

SELECT \*

FROM tesla\_database

LIMIT 10;

-- 2. List all distinct stock names in the dataset

SELECT DISTINCT \*

FROM tesla\_database;

-- 3. Find the date and closing price when Tesla had the highest price.

SELECT `Date`, `Close`, `High`

FROM tesla\_database

WHERE `High` = (SELECT MAX(`High`) FROM tesla\_database);

-- OR

SELECT `Date`, `Close`, `High`

FROM tesla\_database

ORDER BY `High` DESC

LIMIT 1;

-- 4. What is the average closing price of Tesla in 2023?

SELECT

YEAR(`Date`) AS year,

ROUND(AVG(`Close`), 4) AS average\_closing\_price

FROM tesla\_database

WHERE YEAR(`Date`) = 2023

GROUP BY YEAR(`Date`);

-- 5.Count how many trading days are available for each stock.

SELECT

COUNT(DISTINCT `Date`) AS Trading\_Days

FROM tesla\_database;

-- 6. What is the highest and lowest volume traded for stock?

SELECT MAX(VOLUME) AS Max\_Volume, MIN(VOLUME) AS Min\_Volume

FROM tesla\_database;

INTERMEDIATE SOLUTION