Introductory Google BigQuery with Gemini AI
Exercise 1
Questions

1 BigQuery and Gemini reference / documentation

Basics of working with BigQuery via the console UI.

Quick start guide for working with loading and querying data as well as querying public datasets.

Official reference for GoogleSQL, the official SQL dialect for Google BigQuery.

Basic guide to using Gemini to assist writing queries,

2 Lab setup for exercises

We will start off by creating another dataset in the existing project which will hold the table that we will be using for this exercise.

We will create a dataset in the current project with the name: `exercise_dataset`
You can skip this step if you wish, since you will access to the dataset that I create, but you can also choose to create your dataset if you wish.

IMPORTANT NOTE: If you are sharing the same project with me (the trainer), all the datasets that you create will be within in the same project. To ensure that the datasets you create are distinguishable from the datasets of other participants of this workshop, please precede the dataset with your name, for e.g. `peter_exercise_dataset`, `jane_exercise_dataset`, etc (make sure you use underscore and not dashes to separate the words).

We will use the `sampletransactions.csv` file in the `data` subfolder of the downloaded workshop resources to populate the contents of a new table in this new dataset.

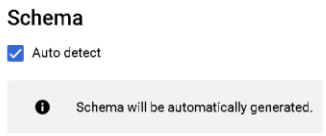
You can open `sampletransactions.csv` in Excel to quickly preview it first if you wish. This file contains dummy data for online trading transactions conducted by several users over several years. The fields / columns in this table are briefly explained as below. Note that the meaning of these fields / columns and their possible values may be different from a real life online trading transaction dataset.

Column	Meaning
`trade_id`	Unique identifier for each trade
`user_id`	Unique identifier for each user. There are currently five users
`platform`	Trading platform. Here, we reference 5 of the main global stock exchanges: NYSE, Nasdaq, SSE (Shanghai Stock Exchange), LSE (London Stock Exchange), Euronext
`currency`	Currency used for the trade, 4 options: USD, EUR, CNY, GBP
`instrument`	Financial instrument being traded. The possible options include: stocks, bonds, ETF, futures, options, CFD, forex, commodities, REITS, mutual
`trade_type`	long or short
`entry_price`	Price at which the order was entered
`exit_price`	Price at which the order was closed
`trade_volume`	Volume or size of the trade
`open_time`	Moment when trader initiates a trade (buying stock, entering forex position, etc). Format: YYYY-MM-DD HH:MM:SS (this aligns with BigQuery's DateTime and also Timestamp data type format, which can optionally include time zone info).
`close_time`	Moment when trade is exited or position is closed (existing forex, selling stock, etc). Format: YYYY-MM-DD HH:MM:SS (this aligns with BigQuery's DateTime and also Timestamp data type format, which can optionally include time zone info).

Create a table in the newly created dataset with the following values in the dialog box that appears.

Create table from: `Upload`
Select file*: `sampletransactions.csv`
File Format: `CSV`
Destination: `intro-bigquery-workshop`
Dataset: `exercise_dataset`
Table: `sampletransactions`
Table type: `Native table`

Tick Auto detect for Schema.

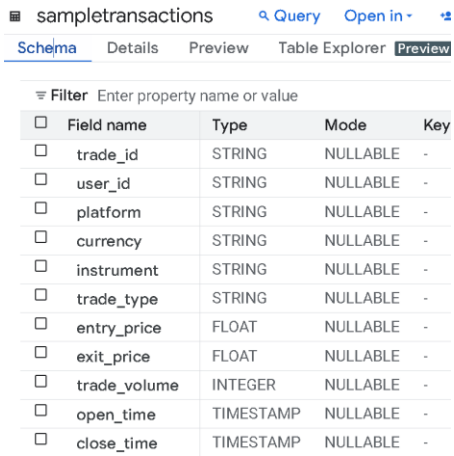


BigQuery will scan the contents of each column and infer the data type for each column as it imports them into the table that it will create.

In Advanced Options, type 1 for Header Rows to skip as the first row in our CSV file is essentially a header row containing the names of the columns/fields for the table we are creating. You can leave the other options as they are.

Finally, click Create Table.
A message about load job running should appear followed by notification about successful creation of the `sampletransactions` table.

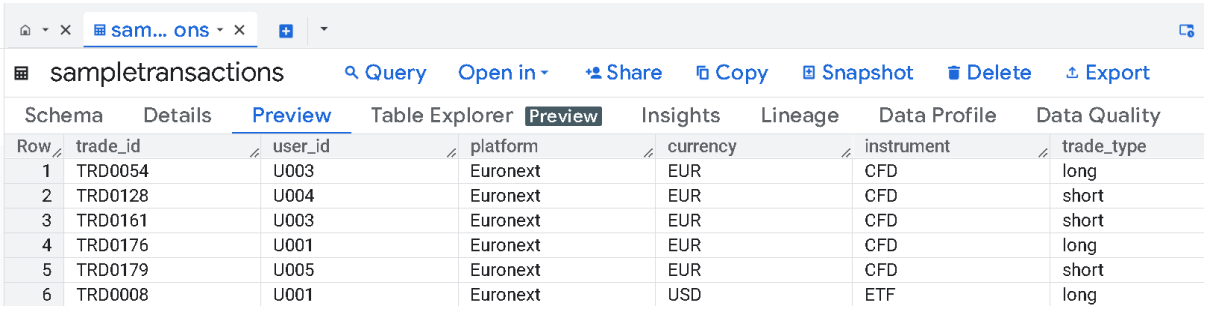
Selecting this table in the Explorer pane should show its Schema in the details pane, where you can see the data types that BigQuery has automatically assigned to each of the fields in the imported CSV file. The Nullable mode indicates this column can contain `null` values (to be covered in a later lab topic)



Field name	Type	Mode	Key
trade_id	STRING	NULLABLE	-
user_id	STRING	NULLABLE	-
platform	STRING	NULLABLE	-
currency	STRING	NULLABLE	-
instrument	STRING	NULLABLE	-
trade_type	STRING	NULLABLE	-
entry_price	FLOAT	NULLABLE	-
exit_price	FLOAT	NULLABLE	-
trade_volume	INTEGER	NULLABLE	-
open_time	TIMESTAMP	NULLABLE	-
close_time	TIMESTAMP	NULLABLE	-

Notice that the data type assigned to `open_time` and `close_time` column is the TIMESTAMP datatype, which optionally can include time zones info.

Click on Preview tab in the Details pane to view the first 50 rows in this table.



Row	trade_id	user_id	platform	currency	instrument	trade_type
1	TRD0054	U003	Euronext	EUR	CFD	long
2	TRD0128	U004	Euronext	EUR	CFD	short
3	TRD0161	U003	Euronext	EUR	CFD	short
4	TRD0176	U001	Euronext	EUR	CFD	long
5	TRD0179	U005	Euronext	EUR	CFD	short
6	TRD0008	U001	Euronext	USD	ETF	long

You will see that the `open_time` and `close_time` column both have the UTC time zone (the default time zone or GMT) assigned to them as no time zone data was specified in the original CSV data file.
If you need to specify time zone for date / time values in Google BigQuery, below are some examples of valid time zone values:

`2025-05-12 09:00:00+08:00` (UTC+8)
`2024-10-01 12:00:00-07:00` (UTC-7)
`2024-08-10 13:00:00 America/New_York` (EST)
`2024-09-11 15:00:00 America/Los_Angeles` (PST)
`2023-08-04 22:00:00 Asia/Shanghai` (CST)
`2023-08-04 22:00:00 Asia/Kuala_Lumpur` (CST)

Notice that the rows in this table (based on the TRDxxx sequences) do not appear in the same sequence as the initial data in `sampletransactions.csv`

This is because the job executed by BigQuery to load the data from this file executes in parallel to populate the table with the data, resulting in rows appearing out of the original sequence, just as in the case of the lab session.

2.1 Prompt template for other AI tools

If you are planning to use other AI tools such as ChatGPT, Grok or Claude for your prompts to generate queries, remember to specify the schema of the table first in this prompt template below:

```
I have created a table called sampletransactions in a dataset called
exercise_dataset in Google BigQuery with the following schema and
data types:

CustID: INTEGER
Date: DATE
FirstName: STRING
LastName: STRING
Region: STRING
State: STRING
```

```
ProdCategory: STRING
Price: FLOAT
Units: INTEGER

Create a GoogleSQL query that ……
```

3 Basic SELECT

Q1 Display the following columns for all rows in the table: `trade_type`, `user_id` and `currency` in that specific order.

Sample result:

Query results

Job information | Results | Chart | JSON | Execution de

Row	trade_type	user_id	currency
1	long	U003	EUR
2	short	U004	EUR
3	short	U003	EUR
4	long	U001	EUR
5	short	U005	EUR
6	long	U001	USD
7	short	U002	USD

3.1 SELECT with expressions and aliases

Q2 In evaluating a transaction, the price difference is computed as the difference between the exit and entry prices for a particular transaction. Compute this value for all rows in the table and give it a meaningful column name: `difference`

Sample result:

Query results

Job information | Results | Chart | JSON | Execution

Row	trade_id	entry_price	exit_price	difference
1	TRD0054	6.51	6.05	-0.459999...
2	TRD0128	2.84	10.0	7.16
3	TRD0161	4.2	5.93	1.7299999...
4	TRD0176	8.21	1.92	-6.290000...
5	TRD0179	4.03	9.38	5.3500000...

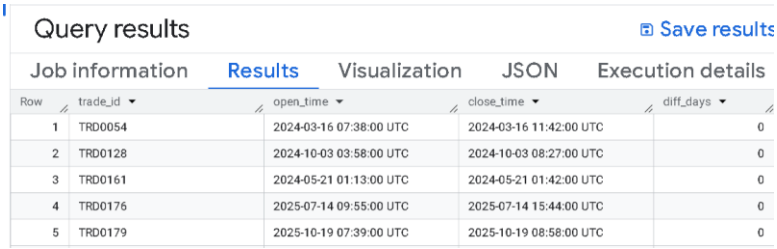
Q3 The trade duration is the length of time a trade is held open, essentially the difference between the closing time (`close_time`) and opening time (`open_time`) of a trade transaction. Compute the duration in terms of total hours or total minutes or total seconds for all rows in the table. You will need to write a separate query to compute the duration for each of these (i.e. total of 3 queries).

HINT: Both columns `close_time` and opening time `open_time` are of the TIMESTAMP data type in Google BigQuery. For this particular data type, BigQuery offers a large number of functions to work

on values from this type. We can use the TIMESTAMP_DIFF function to specify the granularity (DAY, HOUR, MINUTE, SECOND, etc) between closing time (`close_time`) and opening time (`open_time`) For e.g.

```
TIMESTAMP_DIFF(close_time, open_time, DAY)…..
TIMESTAMP_DIFF(close_time, open_time, HOUR)…..
TIMESTAMP_DIFF(close_time, open_time, MINUTE)…..
```

Sample result:

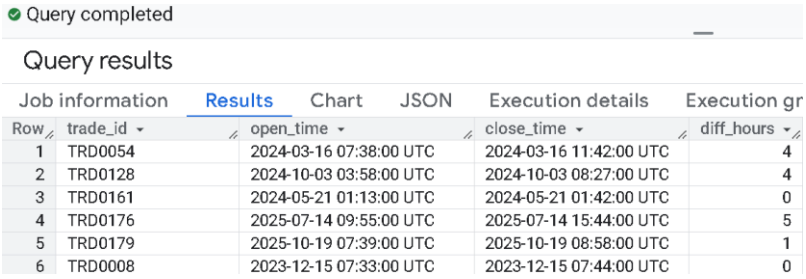


Query results

Save results

Job information | Results | Visualization | JSON | Execution details

Row	trade_id	open_time	close_time	diff_days
1	TRD0054	2024-03-16 07:38:00 UTC	2024-03-16 11:42:00 UTC	0
2	TRD0128	2024-10-03 03:58:00 UTC	2024-10-03 08:27:00 UTC	0
3	TRD0161	2024-05-21 01:13:00 UTC	2024-05-21 01:42:00 UTC	0
4	TRD0176	2025-07-14 09:55:00 UTC	2025-07-14 15:44:00 UTC	0
5	TRD0179	2025-10-19 07:39:00 UTC	2025-10-19 08:58:00 UTC	0

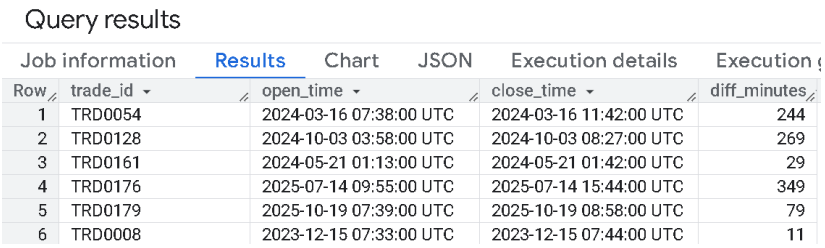


Query completed

Query results

Job information | Results | Chart | JSON | Execution details | Execution gr

Row	trade_id	open_time	close_time	diff_hours
1	TRD0054	2024-03-16 07:38:00 UTC	2024-03-16 11:42:00 UTC	4
2	TRD0128	2024-10-03 03:58:00 UTC	2024-10-03 08:27:00 UTC	4
3	TRD0161	2024-05-21 01:13:00 UTC	2024-05-21 01:42:00 UTC	0
4	TRD0176	2025-07-14 09:55:00 UTC	2025-07-14 15:44:00 UTC	5
5	TRD0179	2025-10-19 07:39:00 UTC	2025-10-19 08:58:00 UTC	1
6	TRD0008	2023-12-15 07:33:00 UTC	2023-12-15 07:44:00 UTC	0



Query results

Job information | Results | Chart | JSON | Execution details | Execution

Row	trade_id	open_time	close_time	diff_minutes
1	TRD0054	2024-03-16 07:38:00 UTC	2024-03-16 11:42:00 UTC	244
2	TRD0128	2024-10-03 03:58:00 UTC	2024-10-03 08:27:00 UTC	269
3	TRD0161	2024-05-21 01:13:00 UTC	2024-05-21 01:42:00 UTC	29
4	TRD0176	2025-07-14 09:55:00 UTC	2025-07-14 15:44:00 UTC	349
5	TRD0179	2025-10-19 07:39:00 UTC	2025-10-19 08:58:00 UTC	79
6	TRD0008	2023-12-15 07:33:00 UTC	2023-12-15 07:44:00 UTC	11

Q4. The previous queries provided the duration in terms of total hours or total minutes or total seconds. This is accurate for mathematical expressions, but may not be so intuitive for human comprehension. Write another query which adds on to your previous queries by using the MOD function in BigQuery to display the duration in terms of both hours and minutes (so for e.g. a duration of 269 minutes is displayed as 4 hours and 29 minutes instead).

Hint: You can nest the `TIMESTAMP_DIFF` function within the `MOD` function so that the result from the `TIMESTAMP_DIFF` function is used by the MOD function.

Sample result:

Query results

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	trade_id	open_time	close_time	hours	minutes
1	TRD0054	2024-03-16 07:38:00 UTC	2024-03-16 11:42:00 UTC	4	4
2	TRD0128	2024-10-03 03:58:00 UTC	2024-10-03 08:27:00 UTC	4	29
3	TRD0161	2024-05-21 01:13:00 UTC	2024-05-21 01:42:00 UTC	0	29
4	TRD0176	2025-07-14 09:55:00 UTC	2025-07-14 15:44:00 UTC	5	49
5	TRD0179	2025-10-19 07:39:00 UTC	2025-10-19 08:58:00 UTC	1	19

Alternatively, you could make it more readable by concatenating results together into a single string.

Query results

Job information | Results | Chart | JSON | Execution details | Execution g

Row	trade_id	open_time	close_time	duration
1	TRD0054	2024-03-16 07:38:00...	2024-03-16 11:42:00...	4 hours 4 minutes
2	TRD0128	2024-10-03 03:58:00...	2024-10-03 08:27:00...	4 hours 29 minutes
3	TRD0161	2024-05-21 01:13:00...	2024-05-21 01:42:00...	0 hours 29 minutes
4	TRD0176	2025-07-14 09:55:00...	2025-07-14 15:44:00...	5 hours 49 minutes
5	TRD0179	2025-10-19 07:39:00...	2025-10-19 08:58:00...	1 hours 19 minutes

3.2 SELECT with DISTINCT and COUNT

Q5. Find all the distinct values possible for the `platform` column:

Sample result:

Query results

Job information | Results

Row	platform
1	Euronext
2	LSE
3	NYSE
4	Nasdaq
5	SSE

Q6. Select all the unique combination of values possible for the columns `currency` and `instrument`.

Sample result:

Query results

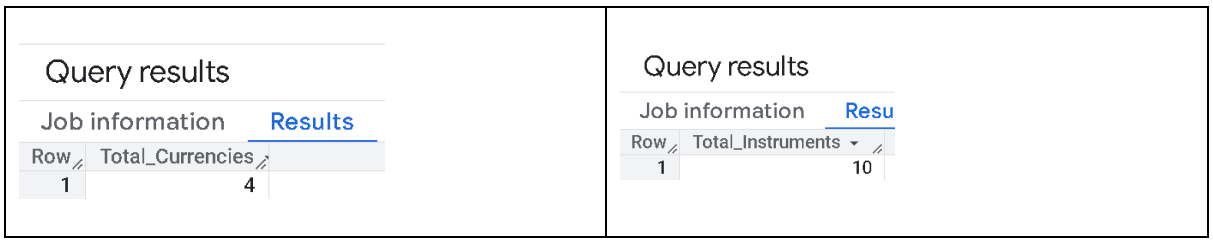
Job information | Results | Chart | JSO

Row	currency	instrument
1	EUR	CFD
2	USD	ETF
3	CNY	REITS
4	USD	bonds
5	GBP	commodities
6	GBP	forex
7	EUR	futures
8	CNY	mutual
9	EUR	options
10	USD	stocks

Notice that there is no repetition of values for the instrument column, which means that each particular category of currency has a set of instruments associated with it that are not found in other currency category. This is important to note when we do hierarchical grouping later on.

Q7. Count how many distinct values are available in the `currency` and `instrument` columns, without viewing these values.

Sample result:



3.3 SELECT with LIMIT

Q8. Show the first 10 rows with all the columns present from this table.

Sample result:

Query results

Save results | Open in

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	trade_id	user_id	platform	currency	instrument	trade_type
1	TRD0054	U003	Euronext	EUR	CFD	long
2	TRD0128	U004	Euronext	EUR	CFD	short
3	TRD0161	U003	Euronext	EUR	CFD	short
4	TRD0176	U001	Euronext	EUR	CFD	long
5	TRD0179	U005	Euronext	EUR	CFD	short
6	TRD0008	U001	Euronext	USD	ETF	long
7	TRD0014	U002	Euronext	USD	ETF	short
8	TRD0034	U003	Euronext	USD	ETF	short
9	TRD0044	U003	Euronext	USD	ETF	short
10	TRD0085	U003	Euronext	USD	ETF	short

4 Using Gemini in BigQuery

5 Sorting rows with ORDER BY

Q1. Sort the rows in ascending order of the `entry_price` and show only the `trade_id` and `entry_price` columns. Limit the result returned to the first 10 rows.

Sample result:

Query results

Job information | Results | Chart

Row	trade_id	entry_price
1	TRD0038	1.03
2	TRD0002	1.04
3	TRD0014	1.13
4	TRD0012	1.22
5	TRD0108	1.25
6	TRD0133	1.27
7	TRD0086	1.28
8	TRD0138	1.33
9	TRD0040	1.38
10	TRD0156	1.49

Q2. Earlier we had computed price difference as the difference between the exit and entry prices for a particular trade. Sort the rows in descending order based on the magnitude of the difference (i.e. we are not interested in the sign + or -, just the absolute value). Limit your result to the first 10 rows.

Hint: Google BigQuery has a large number of mathematical functions we can use in our queries. We can use the ABS function to get the magnitude of a number, regardless of its sign. The queries below are examples:

```
SELECT ABS(10) AS result;
SELECT ABS(-10) AS result;
```

Sample result:

Query results

Job information | Results | Chart | JSON | Execution d

Row	trade_id	entry_price	exit_price	difference
1	TRD0188	9.31	1.26	8.05
2	TRD0160	9.38	1.45	7.93
3	TRD0062	1.63	9.07	7.44
4	TRD0005	9.05	1.63	7.42
5	TRD0086	1.28	8.7	7.42
6	TRD0102	9.53	2.35	7.18
7	TRD0128	2.84	10.0	7.16
8	TRD0164	9.53	2.64	6.89
9	TRD0127	8.1	1.32	6.78
10	TRD0036	2.99	9.31	6.32

Q3. Sort the rows in descending order of the `currency` name. For transactions using the same currency, sort on ascending order of the `trade_volume`.

Sample result:

See file Topic 5 Q3 Results.csv in exercise-solutions.

Q4. Earlier we had computed the trade duration as the difference between the closing time (`close_time`) and opening time (`open_time`) in terms of total hours or total minutes or total seconds. Sort the rows in descending order of the trade duration in total minutes. Limit your result to the first 10 rows.

Sample result:

Query results

Job information | Results | Chart | JSON | Execution details | Exe

Row	trade_id	open_time	close_time	diff_minutes
1	TRD0031	2024-12-13 07:20:00...	2024-12-20 03:58:00...	9878
2	TRD0136	2023-05-25 21:09:00...	2023-06-01 14:33:00...	9684
3	TRD0124	2024-05-24 00:30:00...	2024-05-30 17:16:00...	9646
4	TRD0078	2024-08-16 19:13:00...	2024-08-23 11:15:00...	9602
5	TRD0014	2024-07-04 17:47:00...	2024-07-11 08:26:00...	9519
6	TRD0085	2025-12-03 22:45:00...	2025-12-10 12:16:00...	9451
7	TRD0006	2025-03-16 17:37:00...	2025-03-23 06:26:00...	9409
8	TRD0174	2025-10-20 23:32:00...	2025-10-27 07:58:00...	9146
9	TRD0030	2024-08-07 23:30:00...	2024-08-14 04:23:00...	8933
10	TRD0004	2024-07-14 21:57:00...	2024-07-21 02:40:00...	8923

6 Saving queries, query results and viewing query history

7 Filtering with WHERE

Q1. Identify all the transactions that were made in EUR currency.

Sample result:

Query results

Job information | Results | Chart | JS

Row	trade_id	currency
1	TRD0054	EUR
2	TRD0128	EUR
3	TRD0161	EUR
4	TRD0176	EUR
5	TRD0179	EUR
6	TRD0007	EUR

Q2. Count the total number of transactions which involve the instrument of type `futures`.

Sample result:

Query results

Job information

Row	TotalFutures
1	21

Q3. List all the transactions whose trade volume is more or equals to 3000

Sample result:

See file Topic 7 Q3 Results.csv in exercise-solutions.

Q4. Earlier we had computed price difference as the difference between the exit and entry prices for a particular trade. List all the rows where the price difference is more than 6.0

Sample result:

See file Topic 7 Q4 Results.csv in exercise-solutions.

Q5. Show all the transactions which were not made on the NYSE platform and sort them in descending order based on their `entry_price`. Limit your results to the top 10.

Sample result:

Query results

Job information | Results | Chart | JSON | Exec

Row	trade_id	platform	entry_price
1	TRD0184	LSE	9.97
2	TRD0079	SSE	9.87
3	TRD0047	SSE	9.83
4	TRD0126	Nasdaq	9.7
5	TRD0035	Nasdaq	9.64
6	TRD0046	Nasdaq	9.64
7	TRD0123	SSE	9.63
8	TRD0059	Euronext	9.62
9	TRD0049	SSE	9.53
10	TRD0102	Nasdaq	9.53

Q6. Earlier we had computed the trade duration as the difference between the closing time (`close_time`) and opening time (`open_time`) in terms of total hours or total minutes or total seconds. Find all the trades whose duration is 5 hours or longer, and sort them in descending order. Limit your result to the first 10 rows.

Sample result:

Query results

Job information | Results | Chart | JSON | Execution details | Exe

Row	trade_id	open_time	close_time	diff_hours
1	TRD0031	2024-12-13 07:20:00...	2024-12-20 03:58:00...	164
2	TRD0136	2023-05-25 21:09:00...	2023-06-01 14:33:00...	161
3	TRD0124	2024-05-24 00:30:00...	2024-05-30 17:16:00...	160
4	TRD0078	2024-08-16 19:13:00...	2024-08-23 11:15:00...	160
5	TRD0014	2024-07-04 17:47:00...	2024-07-11 08:26:00...	158
6	TRD0085	2025-12-03 22:45:00...	2025-12-10 12:16:00...	157
7	TRD0006	2025-03-16 17:37:00...	2025-03-23 06:26:00...	156
8	TRD0174	2025-10-20 23:32:00...	2025-10-27 07:58:00...	152
9	TRD0004	2024-07-14 21:57:00...	2024-07-21 02:40:00...	148
10	TRD0030	2024-08-07 23:30:00...	2024-08-14 04:23:00...	148

Q7. Show all the rows where the closing time (`close_time`) and opening time (`open_time`) occur on the same day (YYYY-MM-DD), irrespective of the time of the day.

Hint: You can use the DATE function to return the date portion (YYYY-MM-DD) of the entire time stamp value for both these columns.

Query results

Job information | Results | Chart | JSON | Execution details | Exe

Row	trade_id	open_time	close_time
1	TRD0054	2024-03-16 07:38:00 UTC	2024-03-16 11:42:00 UTC
2	TRD0128	2024-10-03 03:58:00 UTC	2024-10-03 08:27:00 UTC
3	TRD0161	2024-05-21 01:13:00 UTC	2024-05-21 01:42:00 UTC

7.1 Using the AND, OR and NOT operators

Q8. Show the top 10 highest transactions in terms of `trade_volume` that were made in any of these 3 currencies: USD, EUR, GBP. Give two possible alternative forms of the query that you can write.

Sample result:

Query results

Job information | Results | Chart | JSON | Exec

Row	trade_id	currency	trade_volume
1	TRD0045	USD	50000
2	TRD0112	USD	50000
3	TRD0085	USD	50000
4	TRD0140	USD	50000
5	TRD0046	EUR	49000
6	TRD0150	USD	49000
7	TRD0058	GBP	49000
8	TRD0055	EUR	49000
9	TRD0122	EUR	48000
10	TRD0126	USD	48000

Q9. Earlier we had computed price difference as the difference between the exit and entry prices for a particular trade. Show the lowest 3 transactions in terms of this difference for trades that were transacted in GBP on the NYSE.

Sample result:

Query completed

Query results

Job information | Results | Chart | JSON | Execution details | Exec

Row	trade_id	platform	currency	difference
1	TRD0166	NYSE	GBP	0.14
2	TRD0023	NYSE	GBP	0.51
3	TRD0139	NYSE	GBP	4.24

7.2 Using BETWEEN for range tests

Q10. Find all transactions whose trade volume is between 20000 and 40000. Sort your results on the trade volume in descending order.

Sample result:

See file Topic 7 Q10 Results.csv in exercise-solutions.

Q11. List all transactions that took place between June 2024 and June 2025. We consider the transaction to have taken place when it was initiated, not when it closed.

Sample result:

See file Topic 7 Q11 Results.csv in exercise-solutions.

7.3 Using IN to check for matching with other values

Q12. Show the top 10 highest transactions in terms of `trade_volume` that were made in any of these 3 platforms: `NYSE, Nasdaq, LSE`.

Sample result:


Query results

Job information | Results | Chart | JSON | Execu

Row	trade_id	platform	trade_volume
1	TRD0045	LSE	50000
2	TRD0055	Nasdaq	49000
3	TRD0046	Nasdaq	49000
4	TRD0138	Nasdaq	48000
5	TRD0078	NYSE	48000
6	TRD0122	Nasdaq	48000
7	TRD0126	Nasdaq	48000
8	TRD0166	NYSE	47000
9	TRD0017	Nasdaq	47000
10	TRD0116	NYSE	47000

8 Using CASE to implement conditional logic to add columns

Q1. Assume we want to categorize the transactions into 3 categories based on the `trade_volume`.

Trade Volume	Category
10,000 and below	Low
10,001 – 30,000	Medium
30,001 and above	High

Create a new `category` column according to the table above.

Sample result:

See file Topic 8 Q1 Results.csv in exercise-solutions.

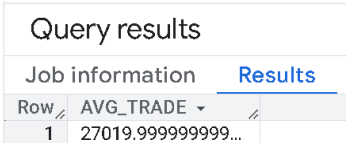
Q2. We will create a new column called `result` which determines whether a particular transaction results in a `profit` or `loss`.
In a long trade (`trade_type = long`), an asset is bought with an expectation that its price will increase – the goal is to buy low at entry and sell high when exiting. For this trade type, a profit occurs when the exit price > entry price.
In a short trade (`trade_type = short`), a borrowed asset is sold with an expectation that its price will fall – the goal is to sell high at entry and buy low when exiting. For this trade type, a profit occurs when the exit price < entry price

Sample result: See file Topic 8 Q2 Results.csv in exercise-solutions.

9 Aggregate functions: COUNT, SUM, AVG, MIN, MAX

Q1. Calculate the average volume of transactions that were performed using the USD currency.

Sample result:

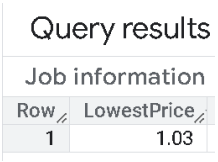

Query results

Job information | Results

Row	AVG_TRADE
1	27019.999999999...

Q2. Find the lowest entry price for all transactions on either the LSE and SSE platform.

Sample result:


Query results

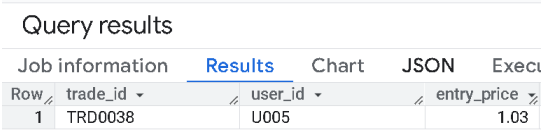
Job information

Row	LowestPrice
1	1.03

9.1 Refining queries with aggregate functions for column details

Q3. Find the `trade_id` and `user_id` for the transaction with the lowest entry price for all transactions on either the LSE and SSE platform
Hint: You can use a subquery from the previous query.

Sample result:


Query results

Job information | Results | Chart | JSON | Execu

Row	trade_id	user_id	entry_price
1	TRD0038	U005	1.03

9.2 Aggregate functions with CASE clause

Q4. Earlier we had categorized the transactions into 3 categories based on the `trade_volume`.

Trade Volume	Category
10,000 and below	Low
10,001 – 30,000	Medium
30,001 and above	High

These 3 new values were placed in a new `category` column.
Find the total number of transactions in each of these categories (Low, Medium and High)

Sample result:

Query results

Job information | Results | Chart

Row	Low	Medium	High
1	37	82	81

10 Aggregating and grouping with GROUP BY

Q1. Count the number of transactions performed in each of the 5 platforms.

Sample result:

Query results

Job information | Results | Char

Row	platform	NumTransca
1	Euronext	48
2	LSE	30
3	NYSE	26
4	Nasdaq	50
5	SSE	46

Q2. Find the transaction with the highest entry price for each currency type. Sort these transactions in descending order based on these entry prices.

Sample result:

Query results

Job information | Results | Chart

Row	currency	HighestPrice
1	GBP	9.97
2	USD	9.87
3	CNY	9.83
4	EUR	9.64

Q3. Find the average trade volume for all long trade transactions for each particular instrument type. Sort these results in descending order of the average trade volume.

Sample result:

Query results

Job information | Results | Chart | JS

Row	instrument	AvgTradeVolume
1	ETF	30375.0
2	commodities	28555.5555555555...
3	options	27571.4285714285...
4	stocks	25750.0
5	futures	25600.0
6	mutual	23600.0
7	REITS	21500.0
8	bonds	18100.0
9	forex	15090.90909090909
10	CFD	11714.2857142857...

10.1 Grouping multiple columns

Q4. Find the lowest exit price of transactions for all groupings of currency and instrument. Order the results in ascending order of this lowest exit price.

Sample result:

Query results

Job information | Results | Visualization | JSON

Row	currency	instrument	LowestPrice
1	GBP	forex	1.02
2	GBP	commodities	1.05
3	EUR	options	1.11
4	EUR	CFD	1.15
5	CNY	mutual	1.15
6	CNY	REITS	1.26
7	USD	ETF	1.34

Q5. Find the total trade volume for short trades on all unique groupings of platform and currency. Your result should show the platforms first with all the currencies associated with that platform listed in descending order of the total trade volume.

Sample result:

Query results

Job information | Results | Visualization | JSON

Row	platform	currency	TotalTrade
1	Euronext	EUR	273000
2	Euronext	USD	248000
3	Euronext	CNY	210000
4	Euronext	GBP	182000
5	LSE	GBP	205000
6	LSE	USD	117000
7	LSE	CNY	108000

10.2 Using HAVING clause to filter on groups

Q6. Find the highest exit price for transactions on all the different instruments. Exclude the instruments whose transaction with the highest exit price is less than 9.5

Sample result:

Query results

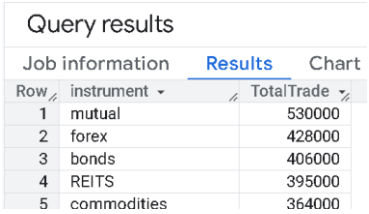
Job information | Results | Chart

Row	instrument	HighestPrice
1	CFD	10.0
2	ETF	9.98
3	REITS	9.86
4	bonds	9.74
5	forex	10.0

Q7. Earlier we have seen that the price difference is computed as the difference between the exit and entry prices for a particular transaction

We want to compute the total trade volume for transactions for all instruments, but exclude transactions whose price difference is 2.0 or less. For the final list, we only want to list instruments whose total trade volume is more than 300,000.

Sample result:



Query results

Job information | Results | Chart

Row	instrument	TotalTrade
1	mutual	530000
2	forex	428000
3	bonds	406000
4	REITS	395000
5	commodities	364000