



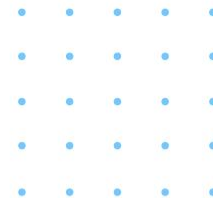
Giới thiệu về Data Warehouse

Data Warehouse Introduction
Advance SQL Query



Khởi nghiệp Game Online

Company Startup



Game Company



**Business Analysis
Squad**

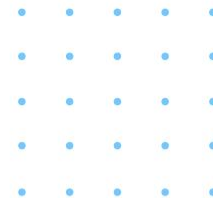


Data Engineer Squad



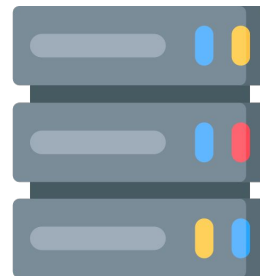
Yêu cầu phân tích

Analytic Request



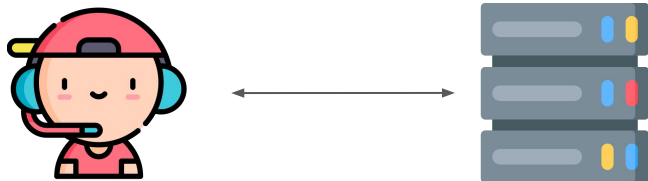
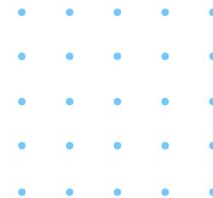
Người chơi

Tương tác



Server





Event ID

Người chơi ID

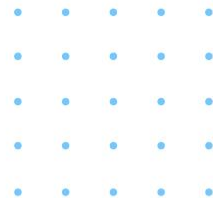
IP

Thiết bị

Thời gian

Loại event

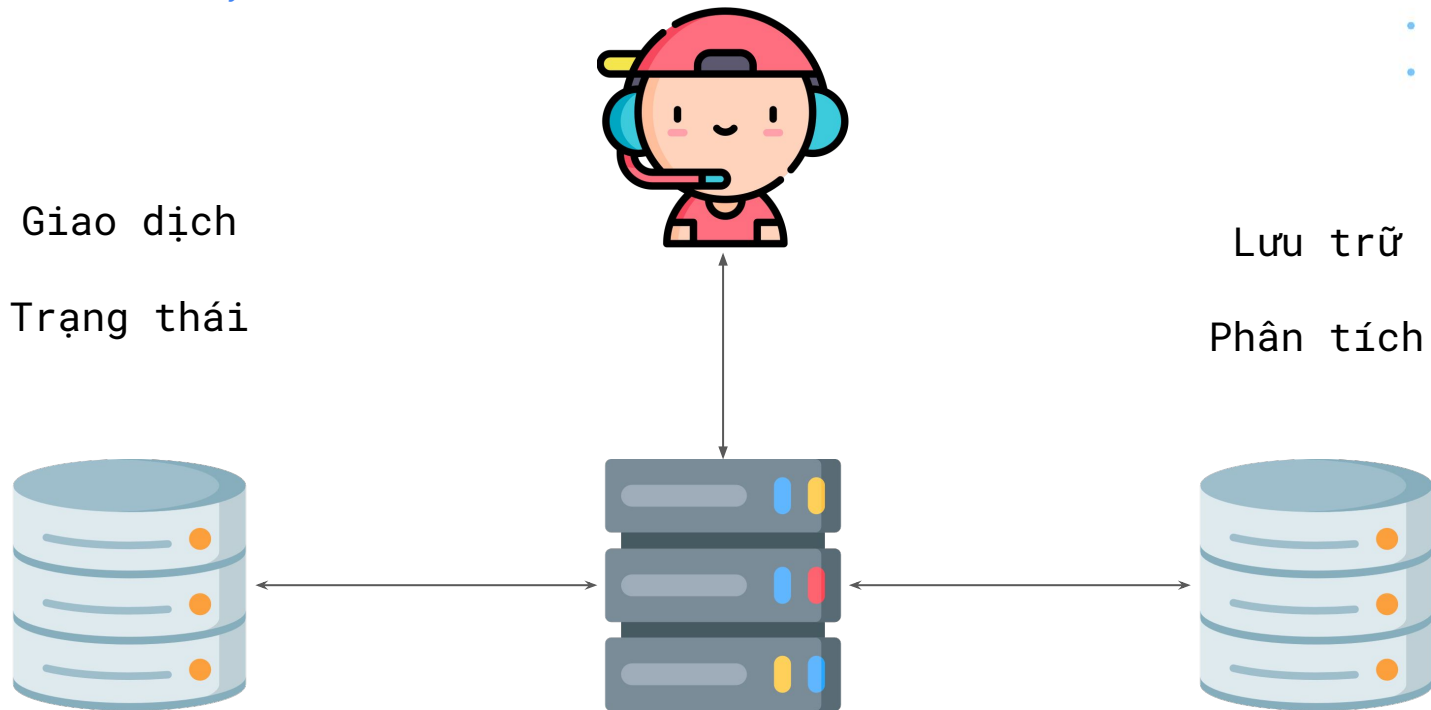
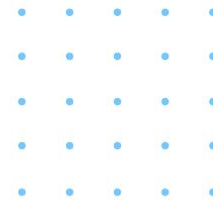
Thuộc tính event



Số lượng **lớn** dữ liệu quá khứ
(event) sẽ được lưu trữ.

Hệ thống không yêu cầu phải
có transaction.





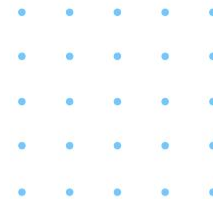
Giao dịch
Trạng thái

Lưu trữ
Phân tích

Quản lý state

Quản lý event





Database

OLTP

Online Transaction Processing



Quản lý state



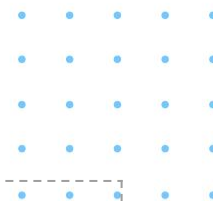
Data Warehouse

OLAP

Online Analytical Processing



Quản lý event

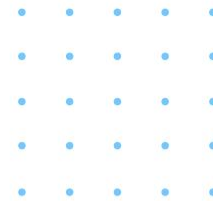


	Database	Data Warehouse
Hệ thống xử lý dữ liệu	OLTP	OLAP
Kiểu dữ liệu	Có cấu trúc	Bán cấu trúc
Tốc độ	Nhanh	Chậm
Cấu trúc lưu trữ	<i>Dòng</i>	<i>Cột</i>
Câu truy hồi	Đơn giản	Phức tạp
Lưu trữ	Trạng thái	Dữ liệu quá khứ
Dùng cho	Quản lý giao dịch	Học máy, phân tích dữ liệu



Hướng dòng so với Hướng cột

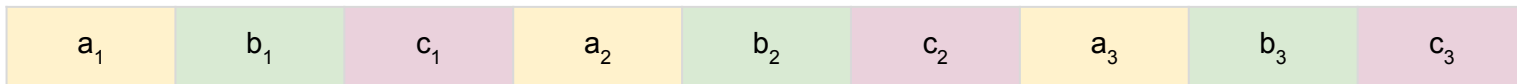
Row-oriented VS Column-oriented



Feature 1	Feature 2	Feature 3
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

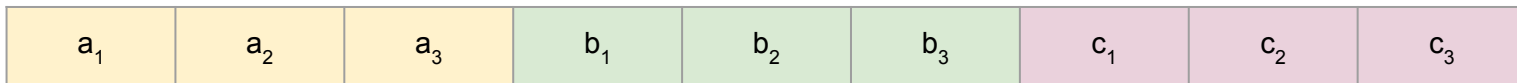
Hướng dòng

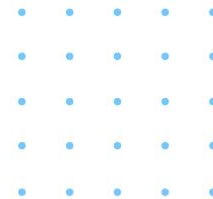
Row-oriented: các dòng đặt liền kề nhau



Hướng cột

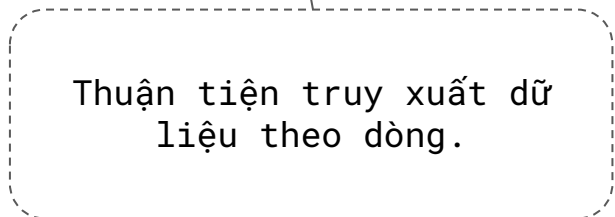
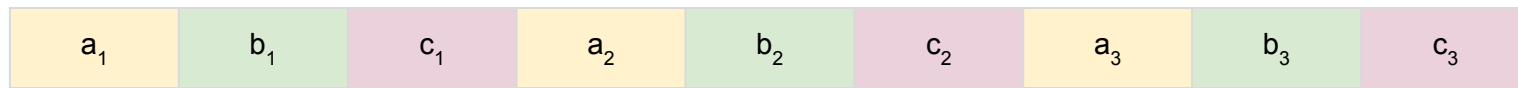
column-oriented: các cột đặt liền kề nhau





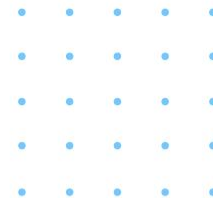
Hướng dòng

Row-oriented: các dòng đặt liền kề nhau



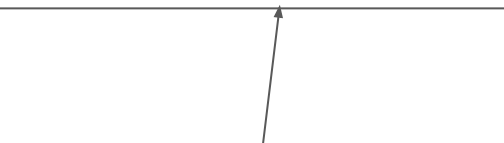
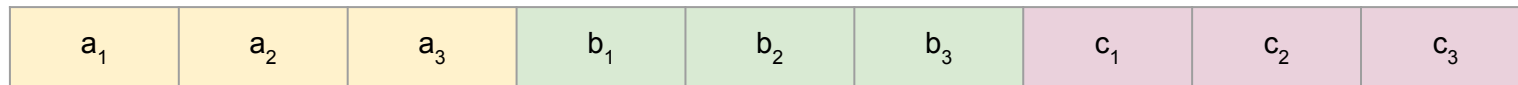
Thuận tiện truy xuất dữ liệu theo dòng.



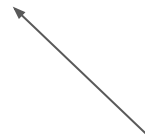


Hướng cột

column-oriented: các cột đặt liền kề nhau



Thuận tiện tổng hợp theo cột.

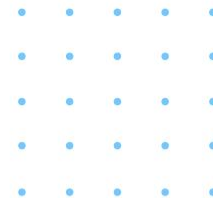


Cùng 1 kiểu dữ liệu nên dễ nén hơn



Data Warehouse

Data Warehouse



BigQuery

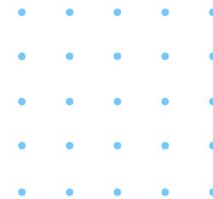


Amazon
Redshift



Giới thiệu BigQuery

Introduction to BigQuery



BigQuery

Sản phẩm của Google Cloud Platform

Serverless Data Warehouse

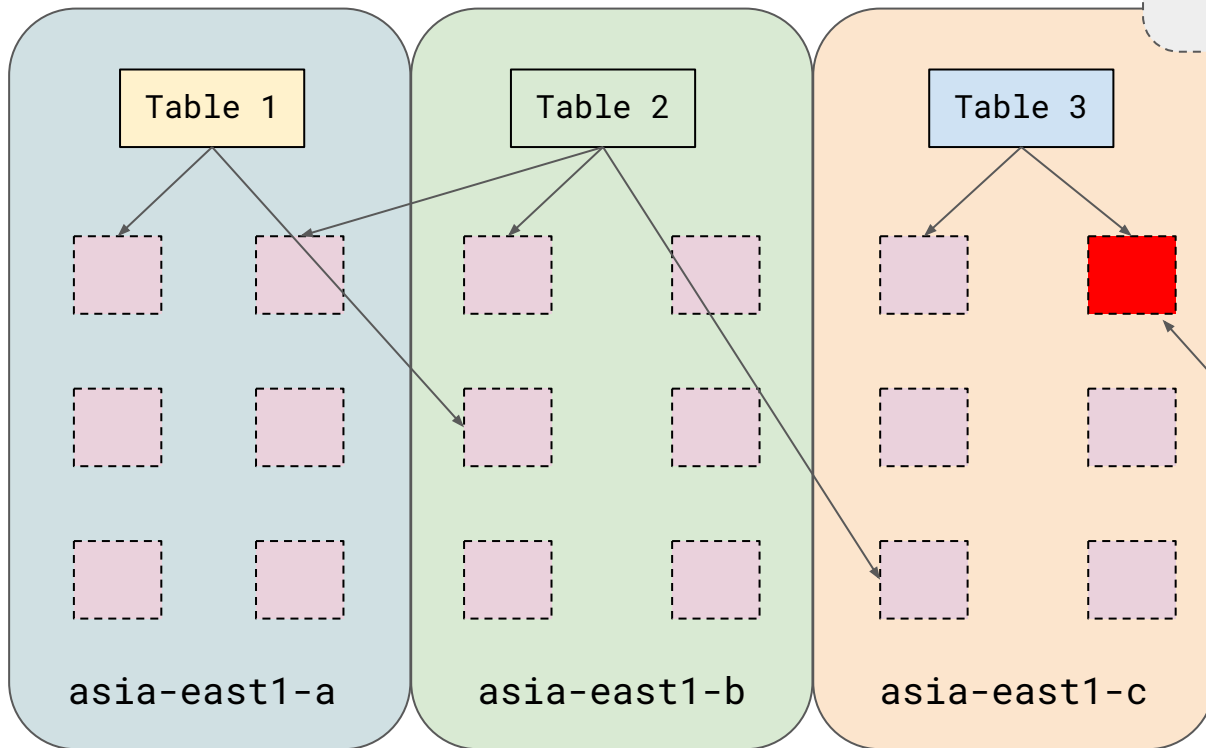
Hỗ trợ cú pháp SQL



Lưu trữ dữ liệu ở BigQuery

How BigQuery Store Data

Region
asia-east1

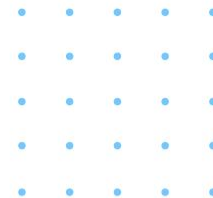


Dữ liệu được chia nhỏ ra, và lưu trữ ở nhiều vùng khác nhau.

Dữ liệu được lưu trữ dư (duplicate) để hạn chế lỗi xảy ra ở vùng lưu trữ

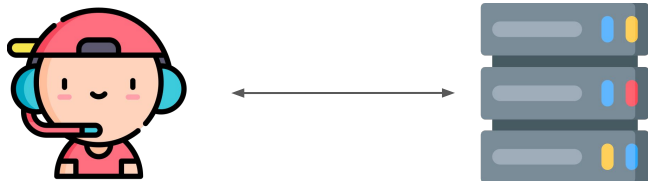
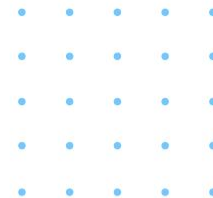
Một số kiểu dữ liệu thường dùng

Data type



Kiểu số	INT64, FLOAT64, NUMERIC
Kiểu đúng sai	BOOL
Kiểu chuỗi	STRING
Kiểu thời gian	DATE, TIMESTAMP, TIME
Kiểu cấu trúc	ARRAY, JSON, STRUCT





Event ID

Người chơi ID

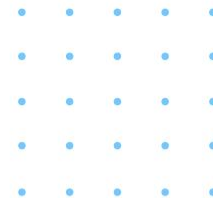
IP

Thiết bị

Thời gian

Loại event

Thuộc tính event



Loại event

purchase

play

view

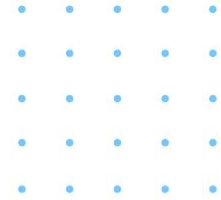


Thuộc tính event

```
"event_attribute":{  
  "revenue" : 90.3,  
  "transaction_id": "ee4c3d7f-e339-4061-be06-80b0a7c2cdcb"  
}
```

```
"event_attribute":{  
  "play_time" : 100  
}
```

```
"event_attribute":{  
  "creative_id" : 1,  
  "view_time": 30,  
  "is_click": false  
}
```



Thuộc tính event

```
"event_attribute":{  
  "revenue" : 90.3,  
  "transaction_id":"ee4c3d7f-e339-4061-be06-80b0a7c2cdcb"  
}
```



```
"event_attribute":[  
  {  
    "key": "revenue",  
    "int_value": None,  
    "Float_value": 90.3,  
    "string_value": None,  
    "bool_value": None,  
  },  
  {  
    "key": "transaction_id",  
    "int_value": None,  
    "float_value": None,  
    "string_value": "ee4c3d7f-e339-4061-be06-80b0a7c2cdcb",  
    "bool_value": None,  
  }  
]
```



Tạo bảng

Create Table

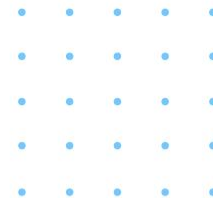
```
CREATE TABLE `adventure_mmo_game.event_log` (  
    event_id STRING,  
    event_type STRING,  
    timestamp TIMESTAMP,  
    user_id INT64,  
    location STRING,  
    device STRING,  
    ip_address STRING,  
    event_attribute ARRAY<STRUCT<  
        key STRING,  
        int_value INT64,  
        float_value FLOAT64,  
        string_value STRING,  
        bool_value BOOL >>);
```

Dùng để lưu trữ dữ liệu
bán cấu trúc JSON

Tạo bảng

Create Table

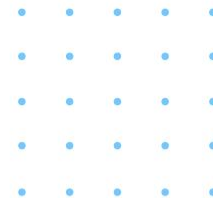
```
CREATE TABLE `adventure_mmo_game.user_info` (  
    user_id INT64,  
    birthday DATE,  
    sign_in_date DATE,  
    sex STRING,  
    country STRING  
);
```

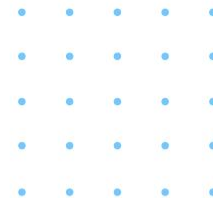




Nhập Data

Import Data





Cú pháp

```
SELECT *  
FROM `<dataset_name>.<table_name>`  
LIMIT NUM_ROW;
```

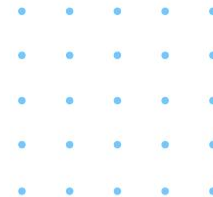


Chỉ lấy NUM_ROW dòng.



Truy hồi

Select



```
SELECT *  
FROM `adventure_mmo_game.event_log`  
LIMIT 1000;
```

Lấy tất cả các cột và 1000
dòng của bảng `event_log`.



Truy hồi

Select

Nên truy xuất tên cột, hạn chế dùng *.

```
SELECT event_id,event_type  
FROM `adventure_mmo_game.event_log`  
LIMIT 1000;
```

Lấy 1000 dòng, cột event_id và event_type của bảng event_log.

Truy hồi

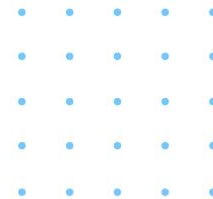
Select

Có thể đặt tên lại cột thông qua câu lệnh **AS**

```
SELECT event_id AS id,  
       event_type AS type  
FROM `adventure_mmo_game.event_log`  
LIMIT 1000;
```

Lấy 1000 dòng, cột event_id và event_type của bảng event_log. Kết quả trả về sẽ là 2 cột có tên id và type

Một số câu lệnh trong Select

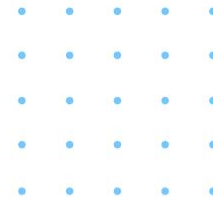


Hàm	Chức năng
<code>DISTINCT</code>	Trả về giá trị độc nhất
<code>CAST</code>	Biến đổi kiểu dữ liệu



Truy hồi

Select



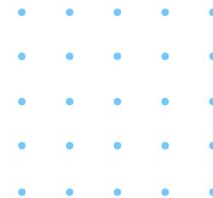
```
SELECT DISTINCT event_type  
FROM `adventure_mmo_game.event_log`  
LIMIT 1000;
```

Lấy giá trị độc nhất
(unique) của event_type.



Truy hồi

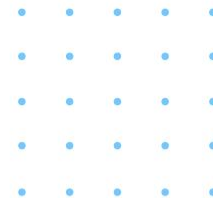
Select



```
SELECT CAST(timestamp AS DATE)
FROM `adventure_mmo_game.event_log`
LIMIT 1000;
```

Đổi kiểu cột timestamp
sang DATE khi trả về





Cú pháp

```
SELECT column_1,column_2  
FROM `<dataset_name>.<table_name>`  
WHERE CONDITIONS  
LIMIT NUM_ROW;
```

Sử dụng câu lệnh **WHERE** kèm theo điều kiện để lọc dữ liệu



Lọc dữ liệu

Where

```
SELECT event_id, event_type
FROM `adventure_mmo_game.event_log`
WHERE (event_type='log_in' OR event_type='log_out')
LIMIT 1000;
```

Lấy cột event_id, event_type.
Lấy những dòng mà event_type có
giá trị là 'log_in' Hoặc
'log_out' của bảng event_log.

Lọc dữ liệu

Where

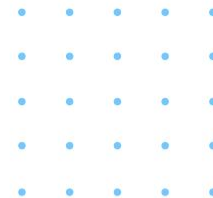
Để query nested array. Ta dùng lệnh **UNNEST**

```
SELECT event_id,event_type,ea.*  
FROM `adventure_mmo_game.event_log`,UNNEST(event_attribute) as ea  
WHERE (ea.key='is_click' AND ea.bool_value)  
LIMIT 1000;
```

Lấy cột event_id, event_type và
tất cả giá trị của
event_attribute có key là
'is_click' VÀ bool_value là **TRUE**

Câu điều kiện

Boolean Expression



Toán tử so sánh

=, <, >, >=, <=, !=

LIKE, BETWEEN, IN, IS

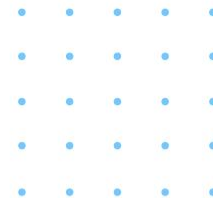
AND, OR

Hàm

STARTWITH, ENDSWITH...

Output của câu điều kiện
phải là kiểu Đúng/Sai





Lấy ra những event mua vật phẩm trong 2 ngày '2023-01-01' và '2023-01-02'.



Lọc dữ liệu

Where

```
SELECT event_id,event_type,timestamp  
FROM `adventure_mmo_game.event_log`  
WHERE event_type LIKE 'purchase'  
AND timestamp BETWEEN '2023-01-01' AND '2023-01-02'  
LIMIT 1000;
```

Lấy ra event_id của những event mua
vật phẩm trong 2 ngày '2023-01-01' và
'2023-01-02'.

Sắp xếp

Order By

Cú pháp

```
SELECT column_1, column_2  
FROM `<dataset_name>.<table_name>`  
WHERE CONDITIONS  
ORDER BY column_1 [ASC | DESC]  
LIMIT NUM_ROW;
```

Tăng dần hoặc giảm dần

Sắp xếp

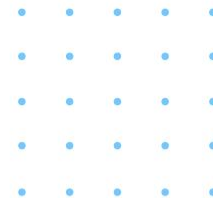
Order By

Cú pháp

```
SELECT column_1, column_2  
FROM `<project_id>.<dataset_name>.<table_name>`  
WHERE CONDITIONS  
ORDER BY 2 [ASC | DESC]  
LIMIT NUM_ROW;
```

Sắp xếp theo cột column_2

Có thể dùng thứ tự của cột trong dòng **SELECT** khi truy hồi



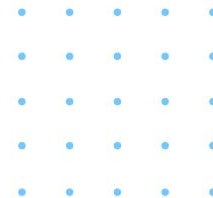
Lấy những event mua vật phẩm mới nhất.

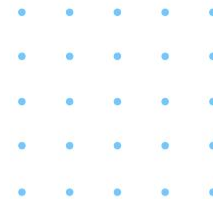


Sắp xếp

Order By

```
SELECT event_id,event_type,timestamp  
FROM `adventure_mmo_game.event_log`  
WHERE event_type LIKE 'purchase'  
ORDER BY timestamp DESC  
LIMIT 1000;
```





Cột cần nhóm phải
có trong câu query

Hàm trả về duy nhất một
giá trị từ một tập hợp
các giá trị.
Ví dụ: Hàm lấy giá trị
nhỏ nhất từ một mảng.

Cú pháp

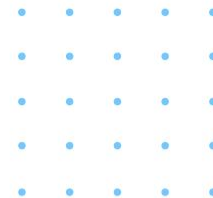
```
SELECT column_1, AGG_FUNCTION(column_2)
FROM `adventure_mmo_game.event_log`
GROUP BY column_1
LIMIT NUM_ROW;
```

Nhóm theo cột 1



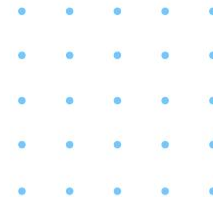
Một số hàm tổng hợp

Aggregate Function



Hàm	Chức năng
MAX	Giá trị lớn nhất
MIN	Giá trị bé nhất
AVG	Giá trị Trung Bình
SUM	Tổng
COUNT	Đếm





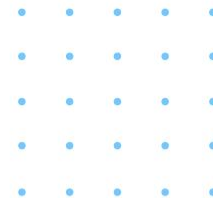
```
SELECT event_type, COUNT(event_id) AS num_row  
FROM `adventure_mmo_game.event_log`  
GROUP BY event_type;
```

Đếm mỗi event_type có bao
nhiêu dòng event_id (bao
nhiêu dòng dữ liệu).



Lọc theo hàm tổng hợp

Having



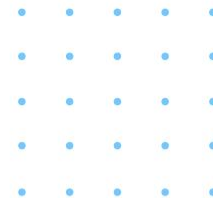
```
SELECT column_1, AGG_FUNCTION(column_2)
FROM `<project_id>.<dataset_name>.<table_name>`
GROUP BY column_1
HAVING CONDITIONS;
```

Sử dụng câu lệnh **HAVING** kèm theo điều kiện để lọc dữ liệu đã qua xử lý ở **AGG_FUNCTION**

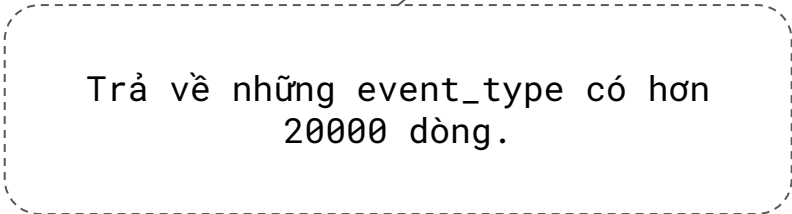


Lọc theo hàm tổng hợp

Having



```
SELECT event_type, COUNT(event_id) AS num_row  
FROM `adventure_mmo_game.event_log`  
GROUP BY event_type  
HAVING COUNT(event_id) >= 20000;
```



Trả về những event_type có hơn
20000 dòng.



Lọc theo hàm tổng hợp

Having

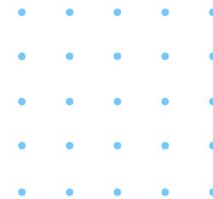


```
SELECT event_type, COUNT(event_id) AS num_row  
FROM `adventure_mmo_game.event_log`  
GROUP BY event_type  
HAVING num_row >= 20000;
```

Trả về những event_type có hơn
20000 dòng.

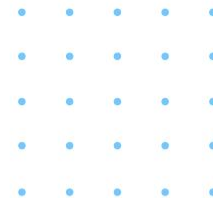
Có thể dùng tên đã đặt lại trong
HAVING





Liệt kê 10 người chơi nào có số lần mua
đồ nhiều nhất ?





```
SELECT e.user_id, COUNT(*) AS total_count
FROM adventure_mmo_game.event_log e
WHERE event_type = 'purchase'
GROUP BY e.user_id
ORDER BY total_count DESC
LIMIT 10;
```



Kết hợp dữ liệu

Join

Có thể lấy cả dữ liệu của cả 2 bảng.

```
SELECT table_1.column_1,  
       table_1.column_2,  
       table_2.column_3,  
       table_2.column_4  
FROM   `<project_id>.<dataset_name>.<table_1_name>` AS table_1  
JOIN   `<project_id>.<dataset_name>.<table_2_name>` AS table_2  
ON     table_1.column_1 = table_2.column_3
```

JOIN để kết hợp 2 bảng lại với nhau theo tiêu chí.

AS để đặt tên 2 bảng lại.

Kết hợp dữ liệu

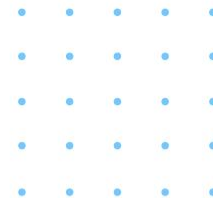
Join

```
SELECT t1.user_id,  
       t1.event_type,  
       t1.location,  
       t2.country,  
       t2.sex  
FROM `adventure_mmo_game.event_log` AS t1  
JOIN `adventure_mmo_game.user_info` AS t2  
ON t1.user_id = t2.user_id  
WHERE t1.event_type='purchase'
```

Lấy dữ liệu của user đã mua hàng và event mua tương ứng.

Kết hợp dữ liệu

Join

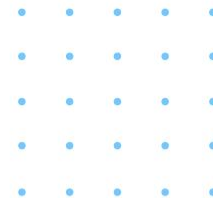


Kiểu Join	Kết quả
JOIN, INNER JOIN	Kết hợp dữ liệu cả hai bên chỉ khi có dữ liệu tương ứng.
LEFT JOIN	Kết hợp dữ liệu từ bảng bên trái và những dữ liệu từ bảng bên phải nếu có dữ liệu tương ứng.
RIGHT JOIN	Kết hợp dữ liệu từ bảng bên phải và những dữ liệu từ bảng bên trái nếu có dữ liệu tương ứng.
OUTER JOIN	Kết hợp dữ liệu cả hai bên. Nếu không có dữ liệu tương ứng dòng đó sẽ hiện NULL .

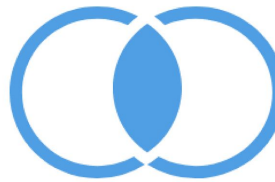


Kết hợp dữ liệu

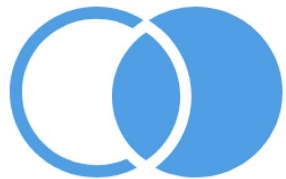
Join



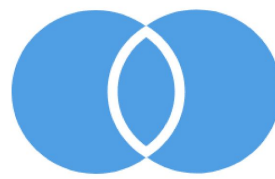
Left outer join



Inner join



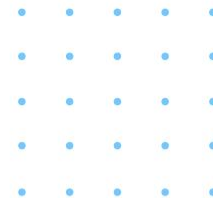
Right outer join



Full outer join



Source: <https://www.metabase.com/learn/sql-questions/sql-join-types>



Thống kê giá trị trung bình, cao nhất,
thấp nhất, số lần mua hàng giữa các nước
?



Query nhỏ

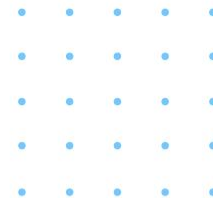
Subquery

```
SELECT column_1,column_2
FROM (
    SELECT column_1,column_2
    FROM `<project_id>.<dataset_name>.<table_name>`
    WHERE CONDITIONS
)
```

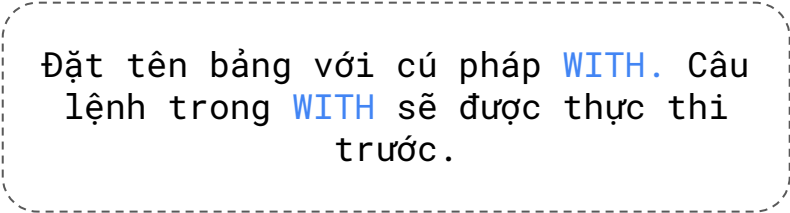
Có thể viết nhiều câu query tạm để
giải quyết các nhu cầu phức tạp hơn

Bảng tạm

Temporary Table



```
WITH temporary_table AS (  
    SELECT column_1, column_2  
    FROM `<project_id>.<dataset_name>.<table_name>`  
    WHERE CONDITIONS  
)  
  
SELECT column_1, column_2  
FROM temporary_table
```

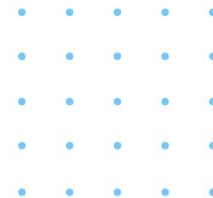


Đặt tên bảng với cú pháp **WITH**. Câu lệnh trong **WITH** sẽ được thực thi trước.



Hàm tổng hợp theo tiêu chí

Window Function



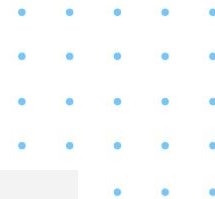
```
SELECT column_1,  
       WINDOW_FUNCTION() OVER (  
         [PARTITION column_1]  
         [ORDER BY column_1]  
         [ROWS BETWEEN FRAM_CONDITION [AND FRAM_CONDITION]  
          | RANGE BETWEEN FRAM_CONDITION [AND FRAM_CONDITION]]]  
       )  
FROM `<project_id>.<dataset_name>.<table_name>`
```

`WINDOW_FUNCTION` là một hàm tổng hợp gom `column_1` theo các tập hợp con chia theo các tiêu chí của `column_1`.



Hàm tổng hợp theo tiêu chí

Window Function



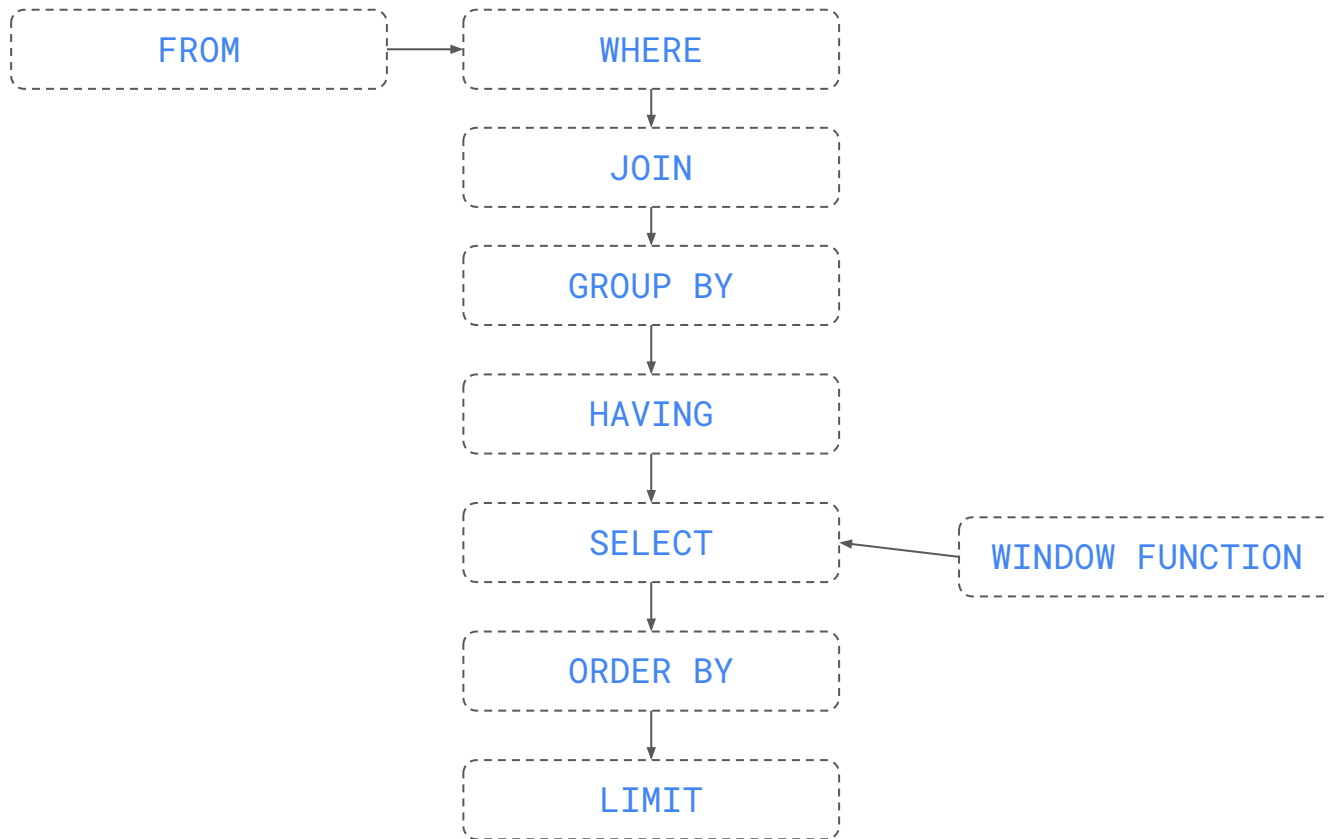
```
WITH LatestEvent AS (  
    SELECT user_id, event_id, event_type, timestamp,  
           ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY timestamp DESC)  
    AS rn  
    FROM `adventure_mmo_game.event_log`  
)  
SELECT user_id, event_id, event_type, timestamp  
FROM LatestEvent  
WHERE rn = 1;
```



Lấy event cuối cùng của mỗi người chơi.

Thứ tự chạy SQL

SQL Executed Order





Data Modeling ảnh hưởng đến chất lượng Query

Cần một số kiến thức của MapReduce để hiểu rõ quy trình xử lý câu query

Tips

Luôn chọn cột cần query. Tránh dùng *

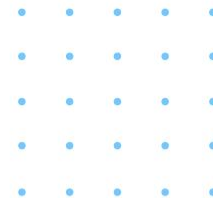
Sử dụng filter được dữ liệu nhất trước

Sử dụng hàm tổng hợp sau cùng

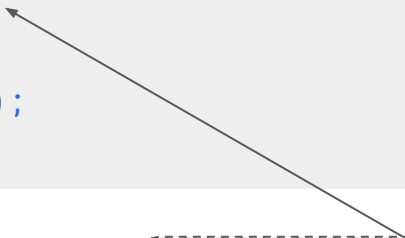
Sử dụng bảng tạm nếu cần thực thi cùng 1 câu lệnh nhiều lần

Sử dụng WHERE để lọc nhiều dữ liệu trước khi JOIN.





```
CREATE TABLE `adventure_mmo_game.event_log` (  
    ...  
)  
PARTITION BY DATE_TRUNC(timestamp,DATE)  
OPTIONS (  
    partition_expiration_days = 3,  
    require_partition_filter = TRUE);  
CLUSTER BY user_id
```



Tạo PARTITION và CLUSTER để
query nhanh hơn

