

COMP3013 Tutorial01

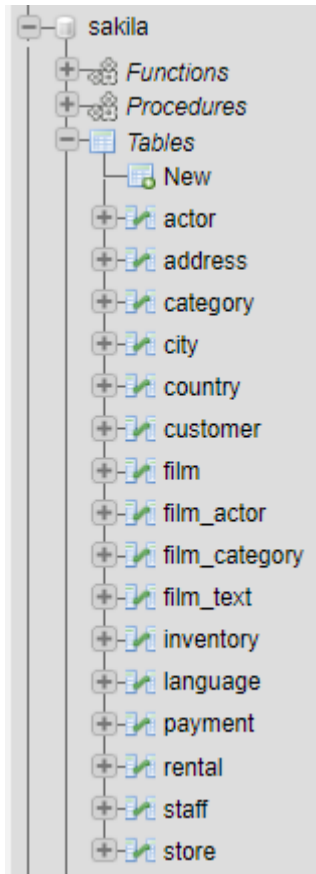
Exercise Solution

2024 Fall

Exercises

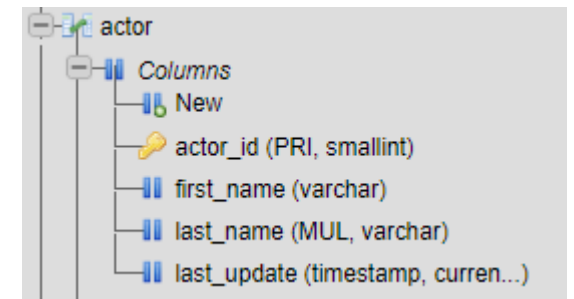
- To write a good query, you need to pay attention to three things.
 1. Which **table** is used?
 2. What is the **predicate**?
 3. What are the **attributes** in the result?
- 1. Find the **information** of **actors** whose **first name** is Russell.
- 2. Find the **information** of **actors** whose **first name** is Russell and **last name** is Close.
- 3. Find the **email** of **customers** whose **first name** is Harry and **active** is 0.
- 4. Find the **full name** of the **actor** whose **id** is 99.
- 5. Find the **title and special features** of the **films** whose **replacement cost** is lower than 20 and **rental rate** is higher than 4.0.
- 6. Find the **customer id** of the customers who have made a **payment** on 2005-05-25 but the **amount** does not exceed 3.

sakila tables



1. Find the **information** of **actors** whose **first name** is **Russell**.

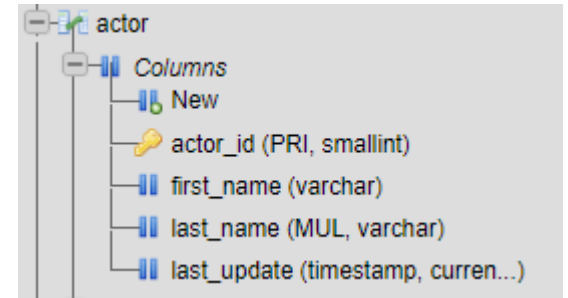
SELECT * FROM ACTOR WHERE FIRST_NAME='RUSSELL'.



			actor_id	first_name	last_name	last_update
Edit	Copy	Delete	112	RUSSELL	BACALL	2006-02-15 04:34:33
Edit	Copy	Delete	149	RUSSELL	TEMPLE	2006-02-15 04:34:33
Edit	Copy	Delete	183	RUSSELL	CLOSE	2006-02-15 04:34:33

2. Find the **information** of **actors** whose **first name** is **Russell** and **last name** is **Close**

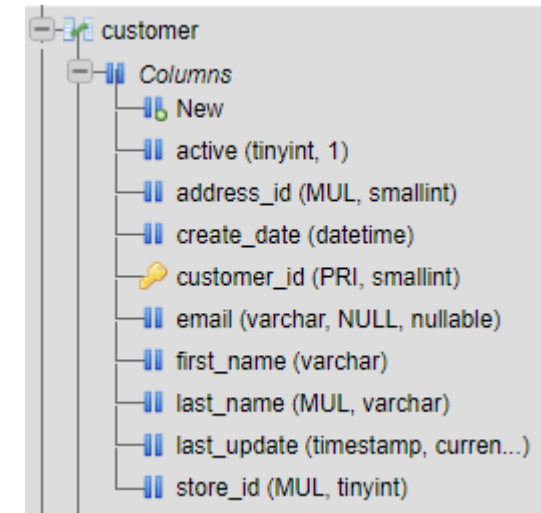
```
SELECT * FROM ACTOR
WHERE
FIRST_NAME='RUSSELL'
AND
LAST_NAME='CLOSE'
```



actor_id	first_name	last_name	last_update
183	RUSSELL	CLOSE	2006-02-15 04:34:33

3. Find the **email** of **customers** whose **first name is Harry and active is 0**.

```
SELECT EMAIL FROM CUSTOMER  
WHERE  
FIRST_NAME='HARRY'  
AND  
ACTIVE=0
```



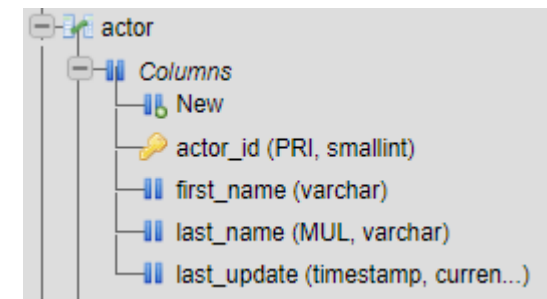
EMAIL

HARRY.ARCE@sakilacustomer.org

4. Find the **full name** of the **actor** whose **id is 99**.

```
SELECT FIRST_NAME, LAST_NAME  
FROM ACTOR  
WHERE ACTOR_ID=99
```

FIRST_NAME	LAST_NAME
JIM	MOSTEL



```
SELECT CONCAT(FIRST_NAME, ' ', LAST_NAME) AS 'FULL NAME'  
FROM ACTOR  
WHERE ACTOR_ID=99
```

FULL NAME
JIM MOSTEL



SQL,查询结果连在一起显示成一列



百度一下

Q 网页 图片 资讯 视频 笔记 地图 贴吧 文库 AI 助手 更多

百度为您找到以下结果

搜索工具

AI 智能回答

SQL,查询结果连在一起显示成一列

如果您想将多个查询结果连在一起显示为一列，可以使用 SQL 的 `CONCAT()` 函数（针对字符串的合并），或者使用 `+` 运算符（在某些数据库中，如 SQL Server），或者使用 `||` 运算符（在 Oracle 或 PostgreSQL 中）。

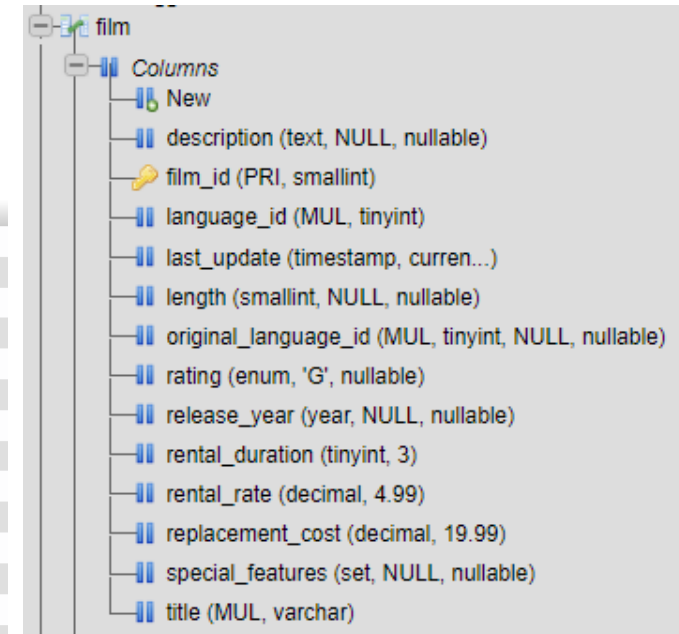
以下是使用 `CONCAT()` 函数的例子，假设我们有两个查询结果 `result1` 和 `result2`：

```
Sql Copy code  
1 SELECT CONCAT(result1, result2) AS combined_result FROM some_table;
```

5. Find the **title and special features** of the **films** whose **replacement cost is lower than 20** and **rental rate is higher than 4.0**.

```
SELECT TITLE, SPECIAL_FEATURES
FROM FILM
WHERE
REPLACEMENT_COST<20
AND
RENTAL_RATE>4.0
```

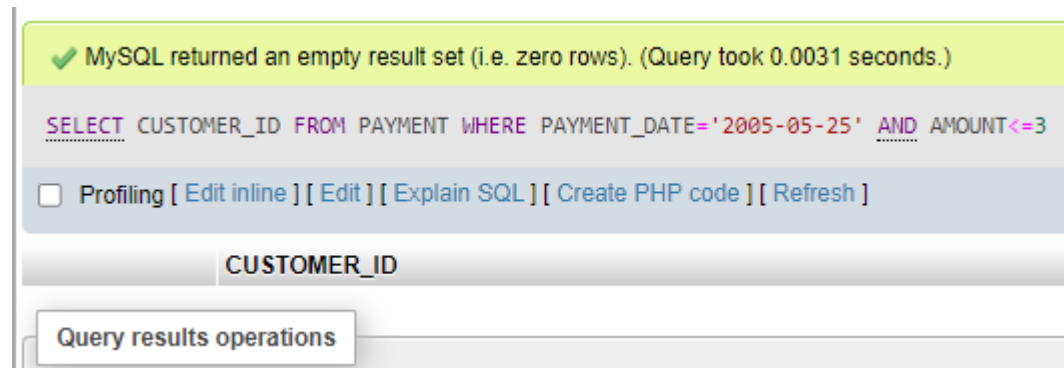
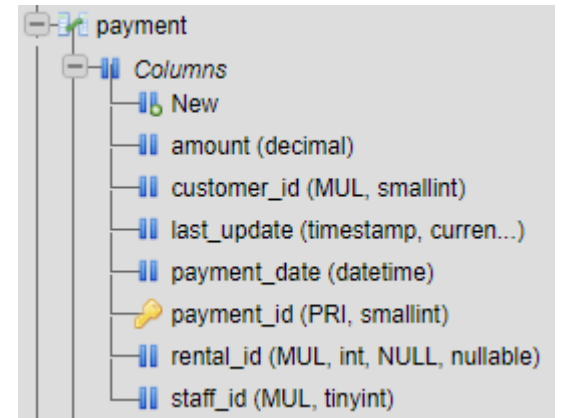
TITLE	SPECIAL_FEATURES
ACE GOLDFINGER	Trailers,Deleted Scenes
AIRPORT POLLOCK	Trailers
AMERICAN CIRCUS	Commentaries,Behind the Scenes
ANTHEM LUKE	Deleted Scenes,Behind the Scenes
APACHE DIVINE	Commentaries,Deleted Scenes,Behind the Scenes
APOCALYPSE FLAMINGOS	Trailers,Commentaries
ATTRACTION NEWTON	Trailers,Behind the Scenes
AUTUMN CROW	Trailers,Commentaries,Deleted Scenes,Behind the Sc...
BIRCH ANTITRUST	Trailers,Commentaries,Deleted Scenes
BIRD INDEPENDENCE	Commentaries,Behind the Scenes
BIRDS PERDITION	Trailers,Behind the Scenes
BOILED DARES	Trailers,Commentaries
BOOGIE AMELIE	Commentaries,Behind the Scenes
BORN SPINAL	Trailers,Commentaries,Deleted Scenes
BOWFINGER GABLES	Trailers,Deleted Scenes
BREAKFAST GOLDFINGER	Trailers,Commentaries,Deleted Scenes
BRIGHT ENCOUNTERS	Trailers
CALIFORNIA BIRDS	Trailers,Commentaries,Deleted Scenes
CANDLES GRAPES	Trailers,Deleted Scenes
CARIBBEAN LIBERTY	Commentaries,Deleted Scenes
CASPER DRAGONFLY	Trailers
CASUALTIES ENCINO	Trailers
CAT CONEHEADS	Commentaries,Deleted Scenes
CENTER DINOSAUR	Deleted Scenes
CHAMBER ITALIAN	Trailers



Columns
New
description (text, NULL, nullable)
film_id (PRI, smallint)
language_id (MUL, tinyint)
last_update (timestamp, current...)
length (smallint, NULL, nullable)
original_language_id (MUL, tinyint, NULL, nullable)
rating (enum, 'G', nullable)
release_year (year, NULL, nullable)
rental_duration (tinyint, 3)
rental_rate (decimal, 4.99)
replacement_cost (decimal, 19.99)
special_features (set, NULL, nullable)
title (MUL, varchar)

6. Find the **customer id** of the customers who have made a **payment** on **2005-05-25** but the **amount does not exceed 3**.

```
SELECT CUSTOMER_ID
FROM PAYMENT
WHERE
PAYMENT_DATE='2005-05-25'
AND
AMOUNT<=3
```



Why?

6. Find the **customer id** of the customers who have made a **payment on 2005-05-25** but the **amount does not exceed 3**.

```
SELECT CUSTOMER_ID
FROM PAYMENT
WHERE
PAYMENT_DATE='2005-05-25'
AND
AMOUNT<=3
```

```
SELECT PAYMENT_DATE
FROM PAYMENT
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0031 seconds.)

```
SELECT CUSTOMER_ID FROM PAYMENT WHERE PAYMENT_DATE='2005-05-25' AND AMOUNT<=3
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

CUSTOMER_ID

Query results operations

Why?

payment

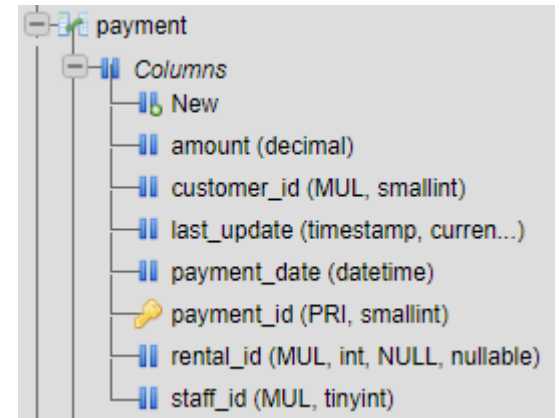
- Columns
- New
- amount (decimal)
- customer_id (MUL, smallint)
- last_update (timestamp, current...)
- payment_date (datetime)
- payment_id (PRI, smallint)
- rental_id (MUL, int, NULL, nullable)
- staff_id (MUL, tinyint)

payment_date
2005-05-25 11:30:37
2005-05-28 10:35:23
2005-06-15 00:54:12
2005-06-15 18:02:53
2005-06-15 21:08:46
2005-06-16 15:18:57
2005-06-18 08:41:48

The data in payment_date includes not only the date but also the payment time, we need to use time range to select out the result

6. Find the **customer id** of the customers who have made a **payment on 2005-05-25** but the **amount does not exceed 3**.

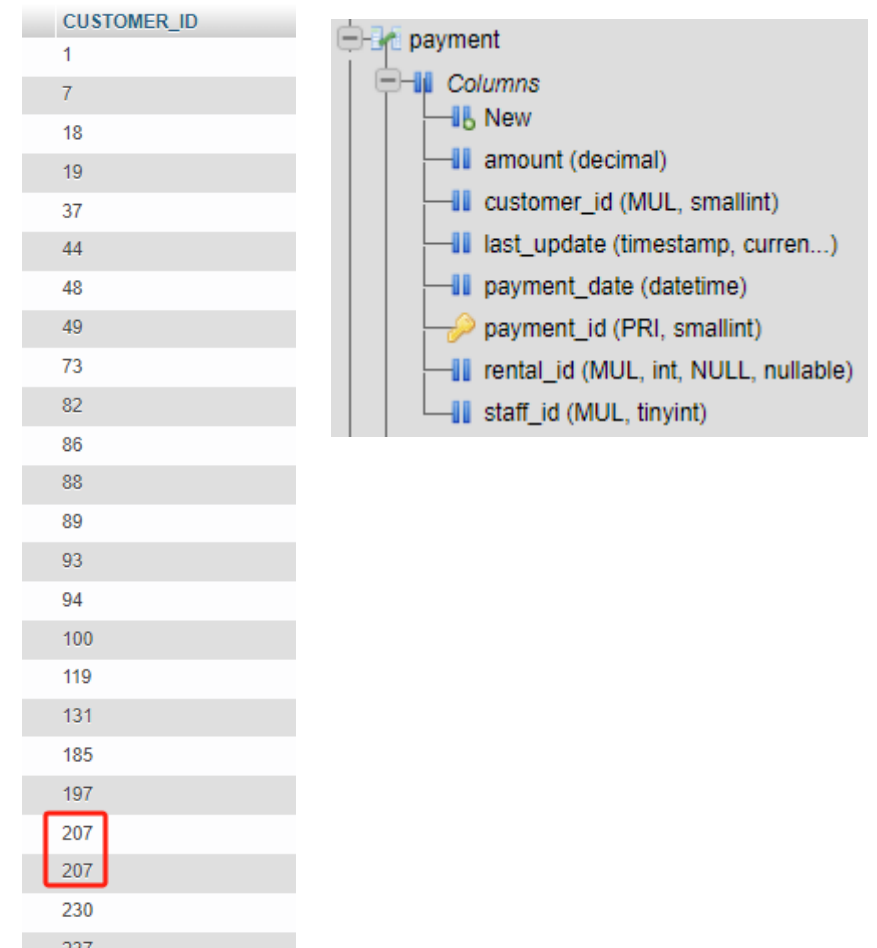
```
SELECT *  
FROM PAYMENT  
WHERE  
PAYMENT_DATE  
BETWEEN '2005-05-25' AND '2005-05-26'  
AND  
AMOUNT<=3
```



payment_id	customer_id	staff_id	rental_id	amount	payment_date	last_update
1	1	1	76	2.99	2005-05-25 11:30:37	2006-02-15 22:12:30
175	7	2	117	0.99	2005-05-25 19:30:46	2006-02-15 22:12:30
468	18	1	50	2.99	2005-05-25 06:44:53	2006-02-15 22:12:32
490	19	2	18	0.99	2005-05-25 01:10:47	2006-02-15 22:12:33
1011	37	1	25	0.99	2005-05-25 03:21:20	2006-02-15 22:12:37
1203	44	1	29	0.99	2005-05-25 03:47:12	2006-02-15 22:12:39
1312	48	2	72	0.99	2005-05-25 10:52:13	2006-02-15 22:12:40
1329	49	2	96	1.99	2005-05-25 16:32:19	2006-02-15 22:12:40

6. Find the **customer id** of the customers who have made a **payment on 2005-05-25** but the **amount does not exceed 3**.

```
SELECT CUSTOMER_ID
FROM PAYMENT
WHERE
PAYMENT_DATE
BETWEEN '2005-05-25' AND '2005-05-26'
AND
AMOUNT<=3
```



The screenshot displays a database interface. On the left, a list of customer IDs is shown, with the value 207 highlighted in a red box. On the right, the schema for the payment table is visible, showing columns: amount (decimal), customer_id (MUL, smallint), last_update (timestamp, current...), payment_date (datetime), payment_id (PRI, smallint), rental_id (MUL, int, NULL, nullable), and staff_id (MUL, tinyint).

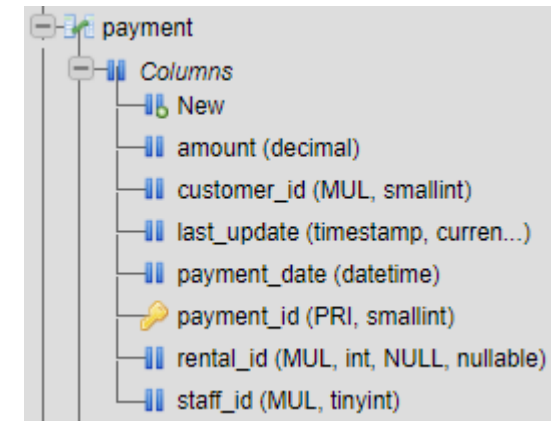
CUSTOMER_ID
1
7
18
19
37
44
48
49
73
82
86
88
89
93
94
100
119
131
185
197
207
207
230
237

payment

- Columns
 - New
 - amount (decimal)
 - customer_id (MUL, smallint)
 - last_update (timestamp, current...)
 - payment_date (datetime)
 - payment_id (PRI, smallint)
 - rental_id (MUL, int, NULL, nullable)
 - staff_id (MUL, tinyint)

6. Find the **customer id** of the customers who have made a **payment on 2005-05-25** but the **amount does not exceed 3**.

```
SELECT DISTINCT CUSTOMER_ID  
FROM PAYMENT  
WHERE  
PAYMENT_DATE  
BETWEEN '2005-05-25' AND '2005-05-26'  
AND  
AMOUNT<=3
```



Use **DISTINCT** to remove duplication

CUSTOMER_ID
1
7
18
19
37
44
48
49
73
82
86
88
89
93
94
100
119
131
185
197
207
230
237
242
257