Project Topic: PMS (Project Management System)

Background & Objective:

A project management system allows retailers to manage their business processes more effectively, ultimately leading to improved customer satisfaction and profitability.

Requirements:

1) the query explanation/description;
2) the SQL statement/commands;
3) the output/result for each query.

Use Cases Overview:

Scenarios	Description	Cases
Senior leadership focus on macro figures	overview of projects	1
Project management focus on project details	workload, staff	2,3,4,5
Inventory department focus on product stock	category, stock	6,7,8
Sales department focus on sales and promotion	sales, discount	9,10,11
Customer department focus on relationship	customer type, register time	12,13,14
Operational research	analysis	15

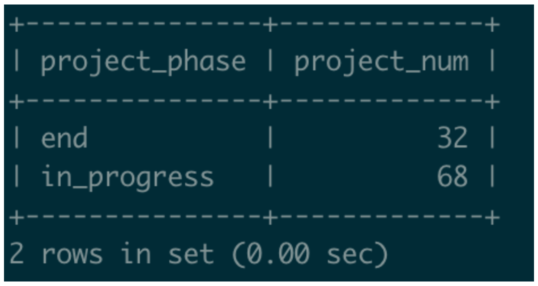
Case Details:

Case1:
The General Manager needs to know the total number of ongoing and closed projects starting this year (2023).

Command1 (General SQL for project table):

```
#the data was generated on 2023-10-6, and this date is used to judge the phase of the project.

select project_phase, count(project_id) as project_num from project where start_date >= '2023-01-01' group by project_phase;
```

Screenshot1:



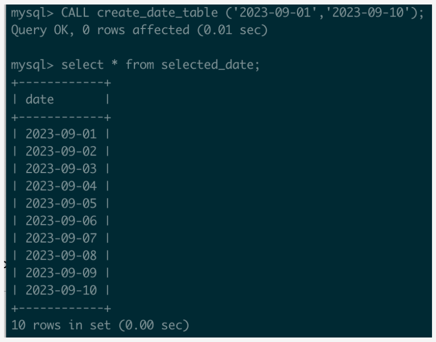
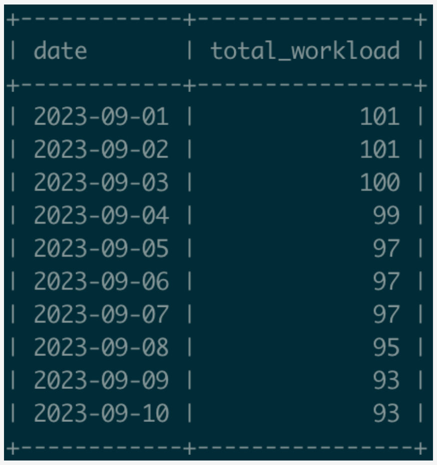
```
+---------------+-------------+
| project_phase | project_num |
+---------------+-------------+
| end           |          32 |
| in_progress   |          68 |
+---------------+-------------+
2 rows in set (0.00 sec)
```

Case2:
The Project Manager needs to keep track of the total in progressing workload of projects between any two dates.

*Command2 (Using **Procedure**):*

```
#the data was generated on 2023-10-6, and this date is used to judge the phase of the project.
#since the data stored in the database is fragmented, a temporary date table needs to be created.
#the whole task will be divided into two parts, (1) creating the date table, (2) obtaining the final result.

#Part1, creating the date table
Delimiter //
drop procedure if exists create_date_table //
create procedure create_date_table (start_date date, end_date date)
begin
  drop table if exists selected_date;
  set @sql = 'create table selected_date ( `date` date not null)';
  prepare stmt from @sql;
  execute stmt;

  while start_date <= end_date do
    insert ignore into selected_date values (date(start_date));
    set start_date = start_date + interval 1 day;
  end while;

  deallocate prepare stmt;

end //
Delimiter ;

CALL create_date_table ('2023-09-01','2023-09-10');

#Part2, obtaining the final result
select date, count(workload) as total_workload from selected_date left join project on date>=start_date
and date<=end_date group by date;
```

Screenshot2:

#Part1	#Part2
	

```
mysql> CALL create_date_table ('2023-09-01','2023-09-10');
Query OK, 0 rows affected (0.01 sec)

mysql> select * from selected_date;
+------------+
| date       |
+------------+
| 2023-09-01 |
| 2023-09-02 |
| 2023-09-03 |
| 2023-09-04 |
| 2023-09-05 |
| 2023-09-06 |
| 2023-09-07 |
| 2023-09-08 |
| 2023-09-09 |
| 2023-09-10 |
+------------+
10 rows in set (0.00 sec)
```

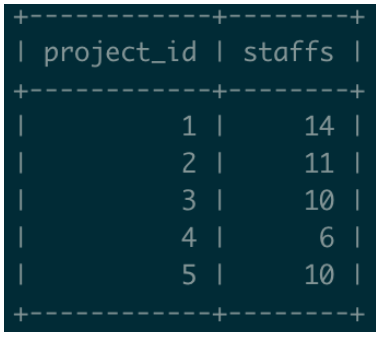
```
+------------+----------------+
| date       | total_workload |
+------------+----------------+
| 2023-09-01 |            101 |
| 2023-09-02 |            101 |
| 2023-09-03 |            100 |
| 2023-09-04 |             99 |
| 2023-09-05 |             97 |
| 2023-09-06 |             97 |
| 2023-09-07 |             97 |
| 2023-09-08 |             95 |
| 2023-09-09 |             93 |
| 2023-09-10 |             93 |
+------------+----------------+
```

Case3:
The Project Manager needs to keep track of the total number of staffs on each project.

*Command3 (Using **Left Join** on **Two** Tables):*

```
select team_project_id  as project_id, count(distinct staff_id) as staffs from task_team t1 left join
task_team_staff t2 on t1.team_task_id = t2.team_task_id group by team_project_id;
```

Screenshot3:



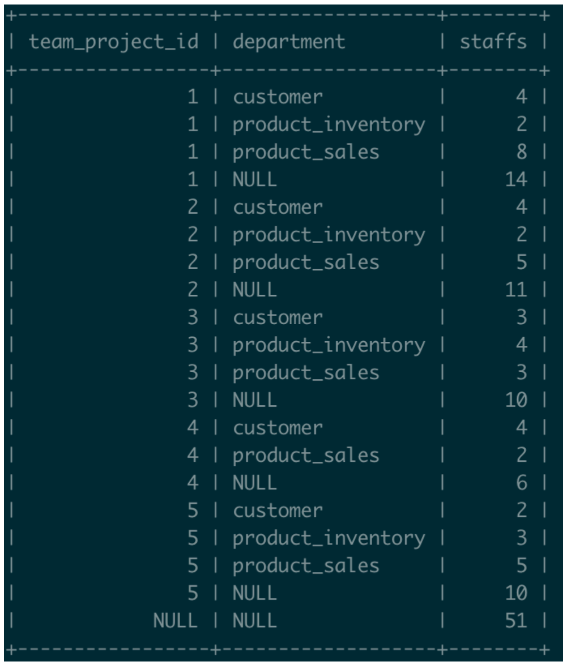
```
+------------+--------+
| project_id | staffs |
+------------+--------+
|          1 |     14 |
|          2 |     11 |
|          3 |     10 |
|          4 |      6 |
|          5 |     10 |
+------------+--------+
```

Case4:
The Project Manager needs to keep track of the total number of staffs in each department on each project, including subtotals and grand total.

*Command4 (Using **Left Join** on **Three** Tables):*

```
select team_project_id, department, count(1) as staffs from task_team t1 left join task_team_staff t2 on
t1.team_task_id=t2.team_task_id left join staff t3 on t2.staff_id=t3.staff_id group by team_project_id,
department with rollup;
```

Screenshot4:



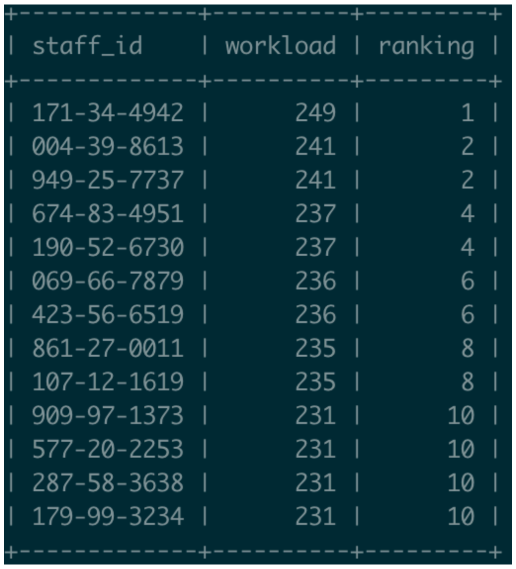
```
+-----------------+-------------------+--------+
| team_project_id | department        | staffs |
+-----------------+-------------------+--------+
|               1 | customer          |      4 |
|               1 | product_inventory |      2 |
|               1 | product_sales     |      8 |
|               1 | NULL              |     14 |
|               2 | customer          |      4 |
|               2 | product_inventory |      2 |
|               2 | product_sales     |      5 |
|               2 | NULL              |     11 |
|               3 | customer          |      3 |
|               3 | product_inventory |      4 |
|               3 | product_sales     |      3 |
|               3 | NULL              |     10 |
|               4 | customer          |      4 |
|               4 | product_sales     |      2 |
|               4 | NULL              |      6 |
|               5 | customer          |      2 |
|               5 | product_inventory |      3 |
|               5 | product_sales     |      5 |
|               5 | NULL              |     10 |
|            NULL | NULL              |     51 |
+-----------------+-------------------+--------+
```

Case5:
The Project Manager needs to rank staffs based on their workload between any two dates and get the top ten ranking result.

*Command5 (Using **Rank()** Function):*

```
# use the procedure created previously

CALL create_date_table ('2023-10-01','2023-10-10');

# get the result

select staff_id, workload, ranking from (select staff_id, workload, rank() over (order by workload desc)
as ranking from (select t2.staff_id, sum(workload) as workload from task_team t1 left join
task_team_staff t2 on t1.team_task_id=t2.team_task_id left join staff t3 on t2.staff_id=t3.staff_id group
by t2.staff_id)t0)t where ranking <=10;
```

Screenshot5:



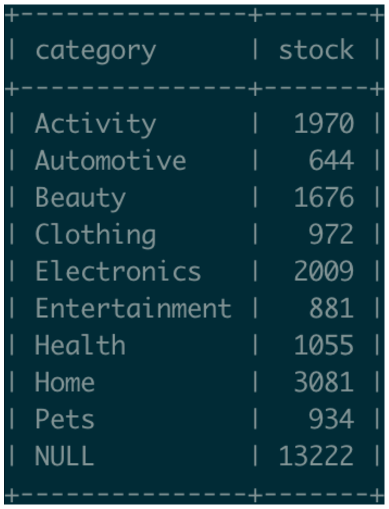
```
+-------------+----------+---------+
| staff_id    | workload | ranking |
+-------------+----------+---------+
| 171-34-4942 |      249 |       1 |
| 004-39-8613 |      241 |       2 |
| 949-25-7737 |      241 |       2 |
| 674-83-4951 |      237 |       4 |
| 190-52-6730 |      237 |       4 |
| 069-66-7879 |      236 |       6 |
| 423-56-6519 |      236 |       6 |
| 861-27-0011 |      235 |       8 |
| 107-12-1619 |      235 |       8 |
| 909-97-1373 |      231 |      10 |
| 577-20-2253 |      231 |      10 |
| 287-58-3638 |      231 |      10 |
| 179-99-3234 |      231 |      10 |
+-------------+----------+---------+
```

Case6:
The head of the inventory department needs to review the inventory for each category, including subtotals and grand total.

*Command6 (Using **OLAP**):*

```
select category, sum(stock_quantity) as stock from product_inventory group by category with rollup;
```

Screenshot6:



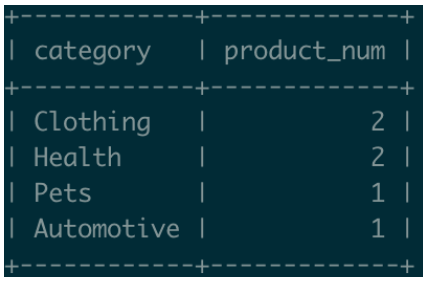
```
+---------------+-------+
| category      | stock |
+---------------+-------+
| Activity      |  1970 |
| Automotive    |   644 |
| Beauty        |  1676 |
| Clothing      |   972 |
| Electronics   |  2009 |
| Entertainment |   881 |
| Health        |  1055 |
| Home          |  3081 |
| Pets          |   934 |
| NULL          | 13222 |
+---------------+-------+
```

Case7:
The head of the inventory department needs to check the number of products that are out of stock.

Command7 (General SQL for product_inventory table):

```
select category, count(product_name) as product_num from product_inventory where
stock_quantity=0 group by category;
```

Screenshot7:



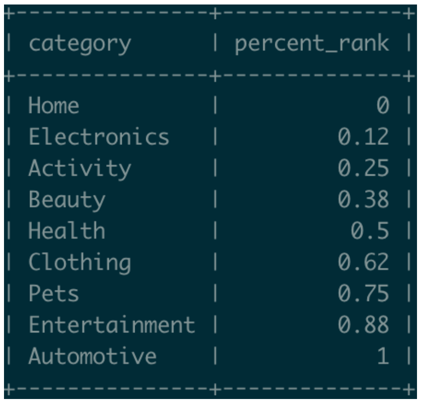
```
+------------+-------------+
| category   | product_num |
+------------+-------------+
| Clothing   |           2 |
| Health     |           2 |
| Pets       |           1 |
| Automotive |           1 |
+------------+-------------+
```

Case8:
The head of the inventory department needs to get the percentage ranking of valid categories.

*Command8 (Using **Percent_Rank()** Function):*

```
select category, round(percent_rank() over (order by stock desc),2) 'percent_rank' from (select
category, sum(stock_quantity) as stock from product_inventory where inventory_status='valid' group
by category)t;
```

Screenshot8:



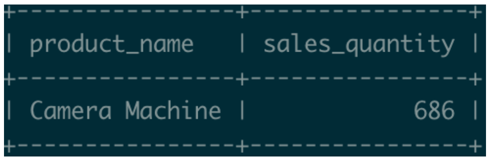
```
+--------------+--------------+
| category     | percent_rank |
+--------------+--------------+
| Home         |            0 |
| Electronics  |         0.12 |
| Activity     |         0.25 |
| Beauty       |         0.38 |
| Health       |          0.5 |
| Clothing     |         0.62 |
| Pets         |         0.75 |
| Entertainment |        0.88 |
| Automotive   |            1 |
+--------------+--------------+
```

Case9:
The head of the sales department needs to know the best-selling product and its sales.

*Command9 (Using **Limit**):*

```
select product_name, sales_quantity from product_sales order by sales_quantity desc limit 1;
```

Screenshot9:



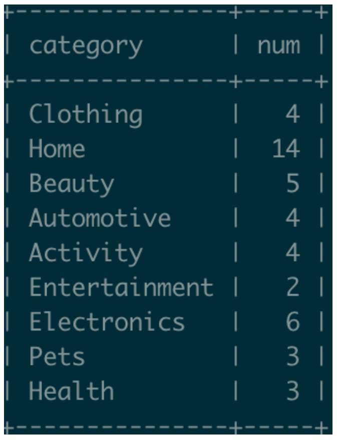
```
+----------------+----------------+
| product_name   | sales_quantity |
+----------------+----------------+
| Camera Machine |            686 |
+----------------+----------------+
```

Case10:
The head of the sales department needs to get the quantity of products with a discount of more than 50%.

Command10 (General SQL for product_sales table):

```
select category, count(product_name) as num from product_sales where discount_rate>=0.5 group by
category;
```

Screenshot10:



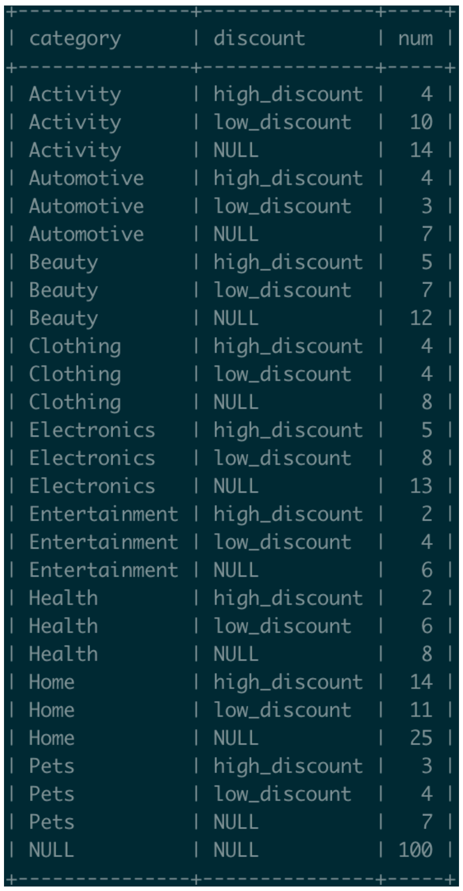
```
+---------------+-----+
| category      | num |
+---------------+-----+
| Clothing      |   4 |
| Home          |  14 |
| Beauty        |   5 |
| Automotive    |   4 |
| Activity      |   4 |
| Entertainment |   2 |
| Electronics   |   6 |
| Pets          |   3 |
| Health        |   3 |
+---------------+-----+
```

Case11:
The head of of sales department needs to obtain the number of products with more than 50% discount or non-discount in each category, including subtotals and grand total.

*Command11 (Using **OLAP**):*

```
select category, case when discount_rate > 0.5 then 'high_discount' else 'low_discount' end as discount, count(product_name) as num from product_sales group by category, case when discount_rate > 0.5 then 'high_discount' else 'low_discount' end with rollup;
```

Screenshot11:



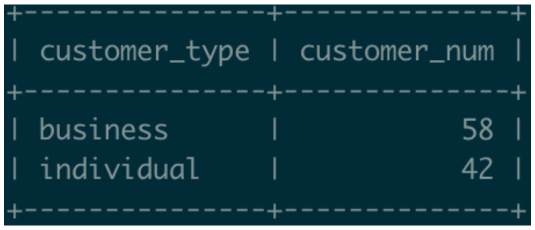
```
+---------------+---------------+-----+
| category      | discount      | num |
+---------------+---------------+-----+
| Activity      | high_discount |   4 |
| Activity      | low_discount  |  10 |
| Activity      | NULL          |  14 |
| Automotive    | high_discount |   4 |
| Automotive    | low_discount  |   3 |
| Automotive    | NULL          |   7 |
| Beauty        | high_discount |   5 |
| Beauty        | low_discount  |   7 |
| Beauty        | NULL          |  12 |
| Clothing      | high_discount |   4 |
| Clothing      | low_discount  |   4 |
| Clothing      | NULL          |   8 |
| Electronics   | high_discount |   5 |
| Electronics   | low_discount  |   8 |
| Electronics   | NULL          |  13 |
| Entertainment | high_discount |   2 |
| Entertainment | low_discount  |   4 |
| Entertainment | NULL          |   6 |
| Health        | high_discount |   2 |
| Health        | low_discount  |   6 |
| Health        | NULL          |   8 |
| Home          | high_discount |  14 |
| Home          | low_discount  |  11 |
| Home          | NULL          |  25 |
| Pets          | high_discount |   3 |
| Pets          | low_discount  |   4 |
| Pets          | NULL          |   7 |
| NULL          | NULL          | 100 |
+---------------+---------------+-----+
```

Case12:
The head of the customer department needs to obtain the number of different types of customers.

Command12 (General SQL for customer table):

```
select customer_type, count(customer_id) from customer group by customer_type;
```

Screenshot12:



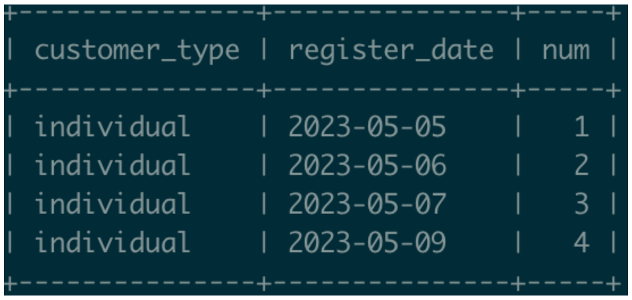
```
+---------------+--------------+
| customer_type | customer_num |
+---------------+--------------+
| business      |           58 |
| individual    |           42 |
+---------------+--------------+
```

Case13:
The head of the customer department needs to obtain the cumulative number of customers for each register_date after ‘2023-05-01’.

*Command13 (Using **Window Function**):*

```
select customer_type, register_date, count(count(customer_id)) over (order by register_date rows unbounded preceding) as num from customer where register_date>='2023-05-01' group by customer_type, register_date;
```

Screenshot13:



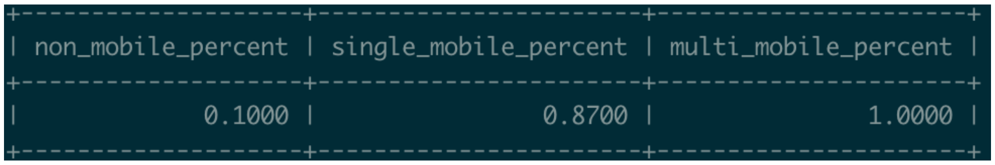
```
+---------------+---------------+-----+
| customer_type | register_date | num |
+---------------+---------------+-----+
| individual    | 2023-05-05    |   1 |
| individual    | 2023-05-06    |   2 |
| individual    | 2023-05-07    |   3 |
| individual    | 2023-05-09    |   4 |
+---------------+---------------+-----+
```

Case14:
The head of the customer department needs to analyze the ratio of the number of customer phone numbers.

*Command14 (Generating **Calculated** Results):*

```
select non_mobile / (total_customer*1.00) as non_mobile_percent, single_mobile / (total_customer*1.00) as single_mobile_percent, multi_mobile / (multi_mobile*1.00) as multi_mobile_percent from (select count(case when mobile=0 then 1 else null end)as non_mobile, count(case when mobile=1 then 1 else null end) as single_mobile, count(case when mobile>1 then 1 else null end) as multi_mobile, count(1) as total_customer from (select t1.customer_id, count(mobile_number) as mobile from customer t1 left join customer_mobile t2 on t1.customer_id=t2.customer_id group by t1.customer_id)t0)t;
```

Screenshot14:



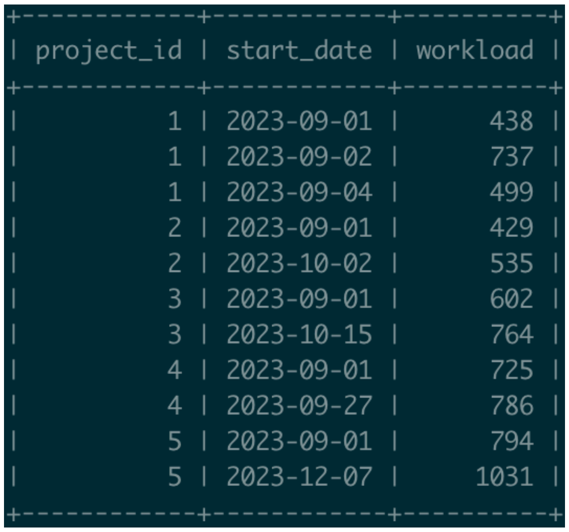
```
+--------------------+-----------------------+----------------------+
| non_mobile_percent | single_mobile_percent | multi_mobile_percent |
+--------------------+-----------------------+----------------------+
|             0.1000 |                0.8700 |               1.0000 |
+--------------------+-----------------------+----------------------+
```

Case15:
Analysts work to find solutions to balance the company's resources, so deviations are checked by taking the total effort curve for two consecutive dates in each project.

*Command15 (Using **Window Function**):*

```
select team_project_id as project_id, start_date, sum(sum(workload)) over (partition by team_project_id order by start_date rows between 1 preceding and current row) as workload from task_team group by team_project_id, start_date order by team_project_id, start_date;
```

Screenshot15:



```
+------------+------------+----------+
| project_id | start_date | workload |
+------------+------------+----------+
|          1 | 2023-09-01 |      438 |
|          1 | 2023-09-02 |      737 |
|          1 | 2023-09-04 |      499 |
|          2 | 2023-09-01 |      429 |
|          2 | 2023-10-02 |      535 |
|          3 | 2023-09-01 |      602 |
|          3 | 2023-10-15 |      764 |
|          4 | 2023-09-01 |      725 |
|          4 | 2023-09-27 |      786 |
|          5 | 2023-09-01 |      794 |
|          5 | 2023-12-07 |     1031 |
+------------+------------+----------+
```