Q1. Order Two Columns Independently

Problem Statement:

Write a query to independently:

- order `first_col` in ascending order.
- order `second_col` in descending order.

SQL Schema:

create table if not Exists data (first_col int, second_col int);

Truncate table data;

insert into data (first_col, second_col) values (4, 2);

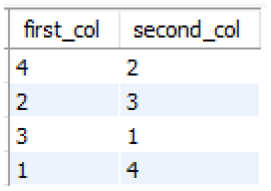
insert into data (first_col, second_col) values (2, 3);

insert into data (first_col, second_col) values (3, 1);

insert into data (first_col, second_col) values (1, 4);

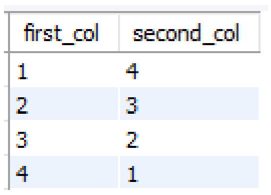
Sample Input:

Table: data



first_col	second_col
4	2
2	3
3	1
1	4

Sample output:



first_col	second_col
1	4
2	3
3	2
4	1

Explanation:

- The first_col has been ordered in ascending manner.
- The second_col has been ordered in descending manner.

Q2. Net Salary

Problem Statement:

Calculate the net salary for the employees and save the column as '**Net_Salary**' and display the details of those employees whose net salary is greater than **15000**.

- Use the CTE method.

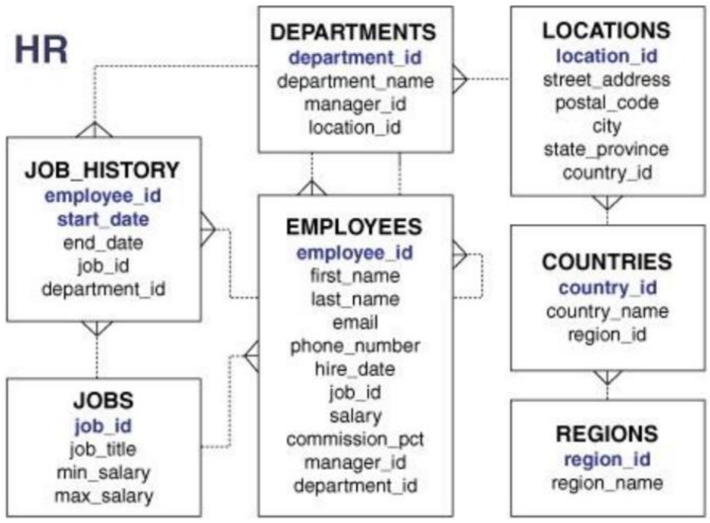
Note: To calculate the `'Net_Salary' = salary + salary *(commission_pct).`

- If the column 'comission_pct' consists of null values replace them with zeros using the ifnull() function.

For example: ifnull(commission_pct,0).

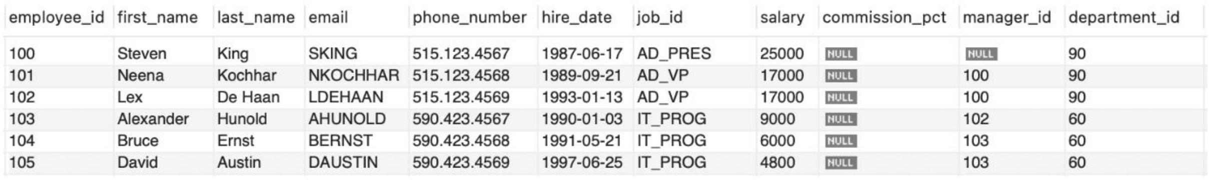
- Return the columns '**employee_id**', '**first_name**', '**last_name**', '**salary**', and '**Net_Salary**'.
- Return the result ordered by **employee_id** in ascending order.

Dataset Description:



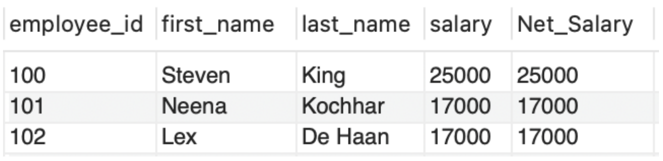
Sample Input:

Table: employees



employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id	department_id
100	Steven	King	SKING	515.123.4567	1987-06-17	AD_PRES	25000	NULL	NULL	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1989-09-21	AD_VP	17000	NULL	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	1993-01-13	AD_VP	17000	NULL	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	1990-01-03	IT_PROG	9000	NULL	102	60
104	Bruce	Ernst	BERNST	590.423.4568	1991-05-21	IT_PROG	6000	NULL	103	60
105	David	Austin	DAUSTIN	590.423.4569	1997-06-25	IT_PROG	4800	NULL	103	60

Sample Output:



employee_id	first_name	last_name	salary	Net_Salary
100	Steven	King	25000	25000
101	Neena	Kochhar	17000	17000
102	Lex	De Haan	17000	17000

Q3. Gender

Problem Description:

Write a SQL query to reorder the entries of the **genders** table so that the "**female**," "**other**," and "**male**" appear in the same order in alternating rows.

The table should be rearranged such that the IDs of each gender are sorted in ascending order.

- Return the column's **user_id** and **gender**.

SQL Schema:

```
create table if not Exists genders (user_id int, gender varchar(20));
Truncate table genders;
insert into genders (user_id, gender) values (1, "male");
insert into genders (user_id, gender) values (2, "other");
insert into genders (user_id, gender) values (3, "other");
insert into genders (user_id, gender) values (4, "male");
insert into genders (user_id, gender) values (5, "female");
insert into genders (user_id, gender) values (6, "male");
insert into genders (user_id, gender) values (7, "female");
```

Sample Input:

Table: genders

user_id	gender
1	male
2	other
3	other
4	male
5	female
6	male
7	female

Sample Output:

user_id	gender
5	female
2	other
1	male
7	female
3	other
4	male

Explanation:

- The output is displayed in this order: "female," "other," and "male" and the ids of each gender are also sorted in ascending order.
- The smallest user_id for female is 5 which is the first row of output
- the next row should be for other, and the smallest user_id is 2.
- The third row should be male for which the smallest user_id is 1.
- This goes on for the remaining data iterating for female other and male

Q4. Most frequently bought

Problem Description :

Write a query to find the **most frequently** ordered product(s) for each customer.

The output should contain the **product_id** and **product_name** for **each customer_id** who ordered **at least one order**.

- Order the output by **customer_id** and **product_id** in ascending order.
- Use the CTE method.

SQL Schema:

Create table If Not Exists Customers (customer_id int, name varchar(10));

Create table If Not Exists Orders (order_id int, order_date date, customer_id int, product_id int);

Create table If Not Exists Products (product_id int, product_name varchar(20), price int);

Truncate table Customers;

insert into Customers (customer_id, name) values ('1', 'Alice');

insert into Customers (customer_id, name) values ('2', 'Bob');

insert into Customers (customer_id, name) values ('3', 'Tom');

insert into Customers (customer_id, name) values ('4', 'Jerry');

insert into Customers (customer_id, name) values ('5', 'John');

Truncate table Orders;

insert into Orders (order_id, order_date, customer_id, product_id) values ('1', '2020-07-31', '1', '1');

insert into Orders (order_id, order_date, customer_id, product_id) values ('2', '2020-7-30', '2', '2');

insert into Orders (order_id, order_date, customer_id, product_id) values ('3', '2020-08-29', '3', '3');

insert into Orders (order_id, order_date, customer_id, product_id) values ('4', '2020-07-29', '4', '1');

insert into Orders (order_id, order_date, customer_id, product_id) values ('5', '2020-06-10', '1', '2');

insert into Orders (order_id, order_date, customer_id, product_id) values ('6', '2020-08-01', '2', '1');

insert into Orders (order_id, order_date, customer_id, product_id) values ('7', '2020-08-01', '3', '3');

insert into Orders (order_id, order_date, customer_id, product_id) values ('8', '2020-08-03', '1', '2');

insert into Orders (order_id, order_date, customer_id, product_id) values ('9', '2020-08-07', '2', '3');

insert into Orders (order_id, order_date, customer_id, product_id) values ('10', '2020-07-15', '1', '2');

Truncate table Products;

insert into Products (product_id, product_name, price) values ('1', 'keyboard', '120');

insert into Products (product_id, product_name, price) values ('2', 'mouse', '80');

insert into Products (product_id, product_name, price) values ('3', 'screen', '600');

insert into Products (product_id, product_name, price) values ('4', 'hard disk', '450');

Sample Input:

```
Customers table:
+-------------+-------+
| customer_id | name  |
+-------------+-------+
| 1           | Alice |
| 2           | Bob   |
| 3           | Tom   |
| 4           | Jerry |
| 5           | John  |
+-------------+-------+
Orders table:
+----------+------------+-------------+------------+
| order_id | order_date | customer_id | product_id |
+----------+------------+-------------+------------+
| 1        | 2020-07-31 | 1           | 1          |
| 2        | 2020-07-30 | 2           | 2          |
| 3        | 2020-08-29 | 3           | 3          |
| 4        | 2020-07-29 | 4           | 1          |
| 5        | 2020-06-10 | 1           | 2          |
| 6        | 2020-08-01 | 2           | 1          |
| 7        | 2020-08-01 | 3           | 3          |
| 8        | 2020-08-03 | 1           | 2          |
| 9        | 2020-08-07 | 2           | 3          |
| 10       | 2020-07-15 | 1           | 2          |
+----------+------------+-------------+------------+
Products table:
+------------+--------------+-------+
| product_id | product_name | price |
+------------+--------------+-------+
| 1          | keyboard     | 120   |
| 2          | mouse        | 80    |
| 3          | screen       | 600   |
| 4          | hard disk    | 450   |
+------------+--------------+-------+
```

Sample Explanation:

- Alice (customer 1) ordered the mouse three times and the keyboard one time, so the mouse is the most frequently ordered product for them.
- Bob (customer 2) ordered the keyboard, the mouse, and the screen one time, so those are the most frequently ordered products for them.
- Tom (customer 3) only ordered the screen (two times), so that is the most frequently ordered product for them.
- Jerry (customer 4) only ordered the keyboard (one time), so that is the most frequently ordered product for them.
- John (customer 5) did not order anything, so we do not include them in the result table.

Q5. Employees with missing info

Problem Statement:

Write a SQL query to report the IDs of all the employees whose information is missing.

The information of an employee is missing if:

- The employee's name is missing, or
- The employee's salary is missing.

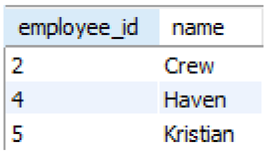
Note: Return the result table ordered by `employee_id` in **ascending order**.

SQL Schema:

```
Create table If Not Exists Employees (employee_id int, name varchar(30));
Create table If Not Exists Salaries (employee_id int, salary int);
Truncate table Employees;
insert into Employees (employee_id, name) values ('2', 'Crew');
insert into Employees (employee_id, name) values ('4', 'Haven');
insert into Employees (employee_id, name) values ('5', 'Kristian');
Truncate table Salaries;
insert into Salaries (employee_id, salary) values ('5', '76071');
insert into Salaries (employee_id, salary) values ('1', '22517');
insert into Salaries (employee_id, salary) values ('4', '63539');
```

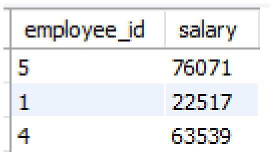
Sample Input:

Table: `employees`



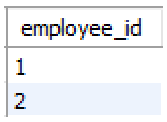
employee_id	name
2	Crew
4	Haven
5	Kristian

Table: `salaries`



employee_id	salary
5	76071
1	22517
4	63539

Sample Output:



employee_id
1
2

Explanation:

- Employees 1, 2, 4, and 5 are working at this company.
- The name of employee 1 is missing.
- The salary of employee 2 is missing.

Q6. Rearranging Products

Problem Statement:

Write a query to rearrange the **Products** table so each row has (**product_id**, **store**, **price**). If a product is **unavailable** in a store, do not include a row with that **product_id** and **store** combination in the result table.

Note:

- Return the result ordered by **product_id** and **store** in ascending order.

SQL Schema:

Create table If Not Exists Products (product_id int, store1 int, store2 int, store3 int);

Truncate table Products;

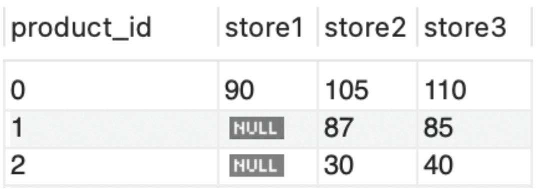
insert into Products (product_id, store1, store2, store3) values ('0', '90', '105', '110');

insert into Products (product_id, store1, store2, store3) values ('1', null, '87', '85');

insert into Products (product_id, store1, store2, store3) values ('2', null, '30', '40');

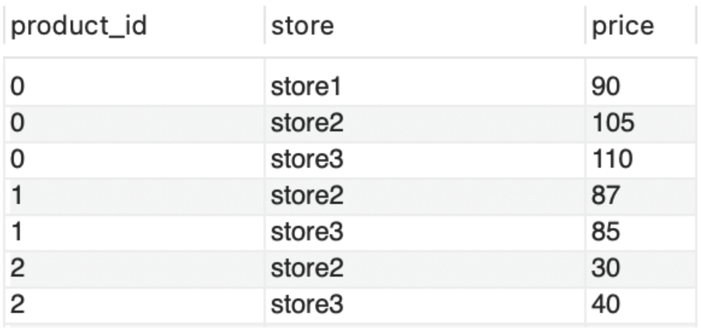
Sample Input:

Table: Products



product_id	store1	store2	store3
0	90	105	110
1	NULL	87	85
2	NULL	30	40

Sample Output:



product_id	store	price
0	store1	90
0	store2	105
0	store3	110
1	store2	87
1	store3	85
2	store2	30
2	store3	40

Sample Explanation:

- Product 0 is available in all three stores with prices 95, 105, and 110 respectively.
- Product 1 is available in store2 with price 87 and store3 with price 85. The product 1 is not available in store1.
- Product 2 is available in store2 with price 30 and store3 with price 40. The product 2 is not available in store1.

Q7. Department name

Problem Description:

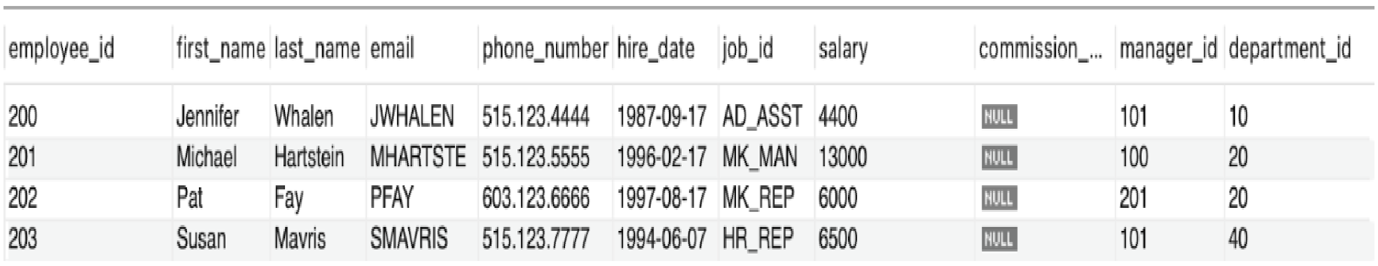
Find the details of the employees who are working in the departments '**Administration**', '**Marketing**', and '**Human Resources**'.

- Return the columns '**employee_id**', '**full_name**'(first and last name separated by space), and '**salary**'.
- Return the result ordered by **employee_id** in ascending order.

Dataset Description is the same as in Q2.

Sample Input:

Table: employees



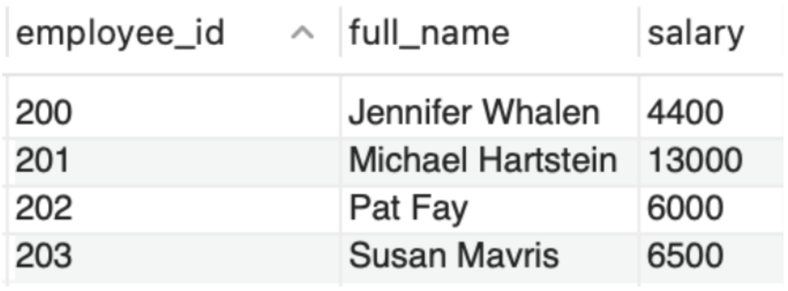
employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_...	manager_id	department_id
200	Jennifer	Whalen	JWHALEN	515.123.4444	1987-09-17	AD_ASST	4400	NULL	101	10
201	Michael	Hartstein	MHARTSTE	515.123.5555	1996-02-17	MK_MAN	13000	NULL	100	20
202	Pat	Fay	PFAY	603.123.6666	1997-08-17	MK_REP	6000	NULL	201	20
203	Susan	Mavris	SMAVRIS	515.123.7777	1994-06-07	HR_REP	6500	NULL	101	40

Table: departments

department_id	department_name	manager_id	location_id
XX	Administration	200	1700
XX	Marketing	201	1800
XX	Human Resources	203	2400

Note: The department_id is hidden for the sample test cases

Sample Output:



employee_id	full_name	salary
200	Jennifer Whalen	4400
201	Michael Hartstein	13000
202	Pat Fay	6000
203	Susan Mavris	6500

Note : Use concat function to add 2 strings.

Q8. Seattle

Problem Statement:

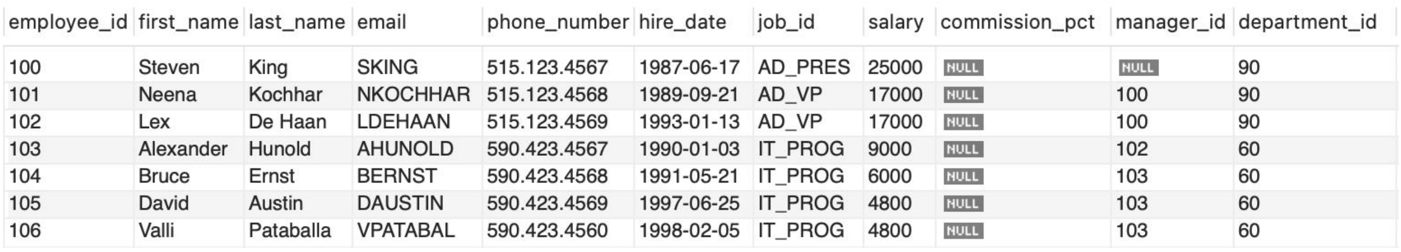
Display the details of all the employees whose department location is in **Seattle**.

- Return the columns '**employee_id**', '**first_name**', '**last_name**', and **job_id**'.
- Return the table ordered by **employee_id** in ascending order.

Dataset Description is the same as Q2

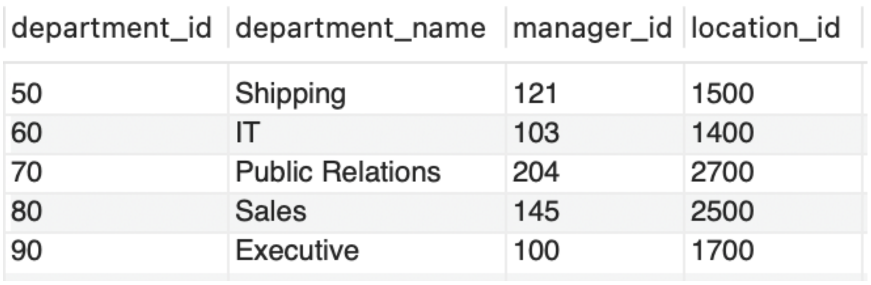
Sample Input:

Table: employees



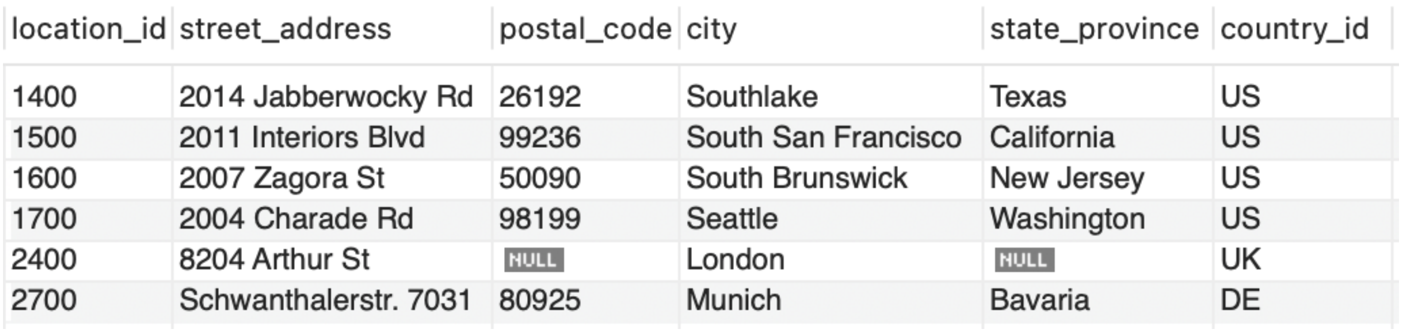
employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id	department_id
100	Steven	King	SKING	515.123.4567	1987-06-17	AD_PRES	25000	NULL	NULL	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1989-09-21	AD_VP	17000	NULL	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	1993-01-13	AD_VP	17000	NULL	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	1990-01-03	IT_PROG	9000	NULL	102	60
104	Bruce	Ernst	BERNST	590.423.4568	1991-05-21	IT_PROG	6000	NULL	103	60
105	David	Austin	DAUSTIN	590.423.4569	1997-06-25	IT_PROG	4800	NULL	103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	1998-02-05	IT_PROG	4800	NULL	103	60

Table: departments



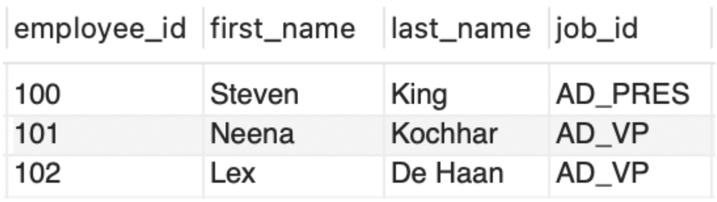
department_id	department_name	manager_id	location_id
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700

Table: locations



location_id	street_address	postal_code	city	state_province	country_id
1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US
1500	2011 Interiors Blvd	99236	South San Francisco	California	US
1600	2007 Zagora St	50090	South Brunswick	New Jersey	US
1700	2004 Charade Rd	98199	Seattle	Washington	US
2400	8204 Arthur St	NULL	London	NULL	UK
2700	Schwanthalerstr. 7031	80925	Munich	Bavaria	DE

Sample Output:



employee_id	first_name	last_name	job_id
100	Steven	King	AD_PRES
101	Neena	Kochhar	AD_VP
102	Lex	De Haan	AD_VP