Q1. Sales Person

Problem Statement:

Write a query to report the names of all the salespersons who did not have any orders related to the company named "**RED**".

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

SQL Schema:

```
Create table If Not Exists SalesPerson (sales_id int, name varchar(255), salary int, commission_rate int, hire_date date);
Create table If Not Exists Company (com_id int, name varchar(255), city varchar(255));
Create table If Not Exists Orders (order_id int, order_date date, com_id int, sales_id int, amount int);
Truncate table SalesPerson;
insert into SalesPerson (sales_id, name, salary, commission_rate, hire_date) values ('1', 'John', '100000', '6', '2006-4-1');
insert into SalesPerson (sales_id, name, salary, commission_rate, hire_date) values ('2', 'Amy', '12000', '5', '2010-5-1');
insert into SalesPerson (sales_id, name, salary, commission_rate, hire_date) values ('3', 'Mark', '65000', '12', '2008-12-25');
insert into SalesPerson (sales_id, name, salary, commission_rate, hire_date) values ('4', 'Pam', '25000', '25', '2005-1-1');
insert into SalesPerson (sales_id, name, salary, commission_rate, hire_date) values ('5', 'Alex', '5000', '10', '2007-2-3');
Truncate table Company;
insert into Company (com_id, name, city) values ('1', 'RED', 'Boston');
insert into Company (com_id, name, city) values ('2', 'ORANGE', 'New York');
insert into Company (com_id, name, city) values ('3', 'YELLOW', 'Boston');
insert into Company (com_id, name, city) values ('4', 'GREEN', 'Austin');
Truncate table Orders;
insert into Orders (order_id, order_date, com_id, sales_id, amount) values ('1', '2014-1-1', '3', '4', '10000');
insert into Orders (order_id, order_date, com_id, sales_id, amount) values ('2', '2014-2-1', '4', '5', '5000');
insert into Orders (order_id, order_date, com_id, sales_id, amount) values ('3', '2014-3-1', '1', '1', '50000');
insert into Orders (order_id, order_date, com_id, sales_id, amount) values ('4', '2014-4-1', '1', '4', '25000');
```

Sample Input:

Table: salesperson

sales_id	name	salary	commission_rate	hire_date
1	John	100000	6	2006-04-01
2	Amy	12000	5	2010-05-01
3	Mark	65000	12	2008-12-25
4	Pam	25000	25	2005-01-01
5	Alex	5000	10	2007-02-03

Table: orders

order_id	order_date	com_id	sales_id	amount
1	2014-01-01	3	4	10000
2	2014-02-01	4	5	5000
3	2014-03-01	1	1	50000
4	2014-04-01	1	4	25000

Table: company

com_id	name	city
1	RED	Boston
2	ORANGE	New York
3	YELLOW	Boston
4	GREEN	Austin

Sample Output:

name
Alex
Amy
Mark

Explanation:

According to orders 3 and 4 in the **orders** table, it is easy to tell that only salespersons John and Pam have sales to company **RED**, so we report all the other names in the table **salesperson**.

Q2. Analysis of Sales

Problem Statement:

Write a query that reports the buyers who have bought the product **S8** but **not** an **iPhone**.

Note:

- Return the result table ordered by **buyer_id** in ascending order.
- **S8** and **iPhone** are products present in the Product table.
- Filters the results based on the **product_name** (i.e., S8, iPhone) as product_id might differ in the test cases.

SQL Schema:

```
Create table If Not Exists Product (product_id int, product_name varchar(10), unit_price int);
Create table If Not Exists Sales (seller_id int, product_id int, buyer_id int, sale_date date, quantity int, price int);
Truncate table Product;
insert into Product (product_id, product_name, unit_price) values ('1', 'S8', '1000');
insert into Product (product_id, product_name, unit_price) values ('2', 'G4', '800');
insert into Product (product_id, product_name, unit_price) values ('3', 'iPhone', '1400');
Truncate table Sales;
insert into Sales (seller_id, product_id, buyer_id, sale_date, quantity, price) values ('1', '1', '1', '2019-01-21', '2', '2000');
insert into Sales (seller_id, product_id, buyer_id, sale_date, quantity, price) values ('1', '2', '2', '2019-02-17', '1', '800');
insert into Sales (seller_id, product_id, buyer_id, sale_date, quantity, price) values ('2', '1', '3', '2019-06-02', '1', '800');
insert into Sales (seller_id, product_id, buyer_id, sale_date, quantity, price) values ('3', '3', '3', '2019-05-13', '2', '2800');
```

Sample Input:

Table: Sales

seller_id	product_id	buyer_id	sale_date	quantity	price
1	1	1	2019-01-21	2	2000
1	2	2	2019-02-17	1	800
2	1	3	2019-06-02	1	800
3	3	3	2019-05-13	2	2800

Table: Product

product_id	product_name	unit_price
1	S8	1000
2	G4	800
3	iPhone	1400

Sample Output:

buyer_id
1

Explanation:

- The buyer with id 1 bought an S8 but did not buy an iPhone.
- The buyer with id 3 bought both.

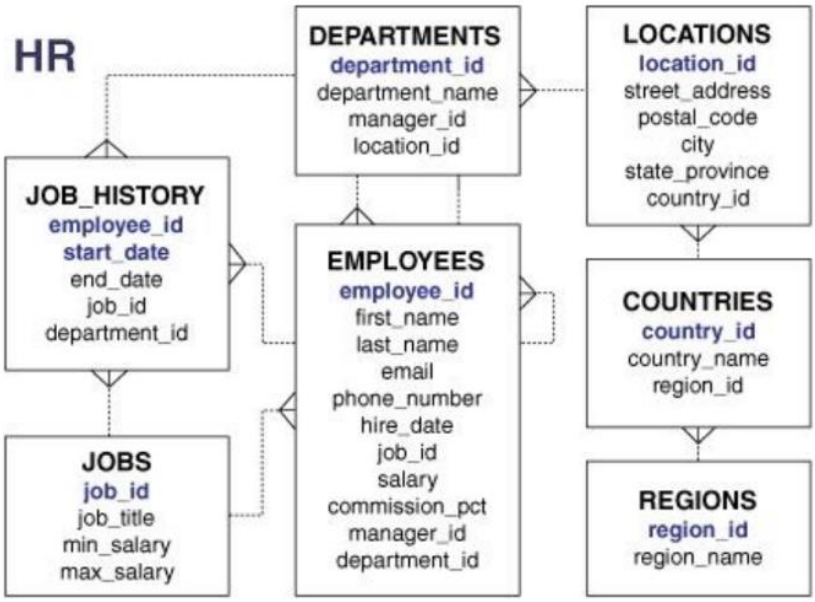
Q3. Adam

Problem Statement:

Write a query to display the list of employees who report to the manager **Adam**. The reporting structure can be inferred from the **manager_id** column in the employee's table.

- Return the columns '**employee_id**', '**full_name**'(first name and last name separated by space), and '**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
121	Adam	Fripp	AFRIPP	650.123.2234	1997-04-10	ST_MAN	8200	NULL	100	50
132	TJ	Olson	TJOLSON	650.124.8234	1999-04-10	ST_CLERK	2100	NULL	121	50
184	Nandita	Sarchand	NSARCHAN	650.509.1876	1996-01-27	SH_CLERK	4200	NULL	121	50
185	Alexis	Bull	ABULL	650.509.2876	1997-02-20	SH_CLERK	4100	NULL	121	50
186	Julia	Dellinger	JDELLING	650.509.3876	1998-06-24	SH_CLERK	3400	NULL	121	50
187	Anthony	Cabrio	ACABRIO	650.509.4876	1999-02-07	SH_CLERK	3000	NULL	121	50
188	Kelly	Chung	KCHUNG	650.505.1876	1997-06-14	SH_CLERK	3800	NULL	122	50
189	Jennifer	Dilly	JDILLY	650.505.2876	1997-08-13	SH_CLERK	3600	NULL	122	50
190	Timothy	Gates	TGATES	650.505.3876	1998-07-11	SH_CLERK	2900	NULL	122	50

- The **manager_id** in the employee's table is the **employee_id** of the manager.

Sample Output:

employee_id	full_name	salary
132	TJ Olson	2100
184	Nandita Sarchand	4200
185	Alexis Bull	4100
186	Julia Dellinger	3400
187	Anthony Cabrio	3000

Q4. Employee 107

Problem Statement:

Write a query to find the details of the other employees who work in the **same job** as the employee with **employee_id** as **107**.

Note:

- Create a new column "**full_name**" by concatenating the **first_name** and **last_name** columns, separated by **space**.
- Return the columns '**full_name**', '**salary**', '**department_id**', and '**job_id**'.
- Return the output ordered by **full_name** in ascending order.

Dataset Description is exactly same as the Q3.

Sample Input:

Table: employees

employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id	department_id
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1989-09-21	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
107	Diana	Lorentz	DLORENTZ	590.423.5567	1999-02-07	IT_PROG	4200	NULL	103	60
115	Alexander	Khoo	AKHOO	515.127.4562	1995-05-18	PU_CLERK	3100	NULL	114	30
151	David	Bernstein	DBERNSTE	011.44.1344.345268	1997-03-24	SA_REP	9500	0.25	145	80
165	David	Lee	DLEE	011.44.1346.529268	2000-02-23	SA_REP	6800	0.1	147	80

- The column **manager_id** in the employees table represents the employee_id of the manager.

Sample Output:

full_name	salary	department_id	job_id
Alexander Hunold	9000	60	IT_PROG
Bruce Ernst	6000	60	IT_PROG
David Austin	4800	60	IT_PROG
Diana Lorentz	4200	60	IT_PROG

Sample Explanation:

Here the employee with id 107 has the job_id as '**IT_PROG**'. In the sample data, we have 4 employees with the same job_id including employee 107.
Hence, we return those 4 records with the same job_id as 107.

Note : Use concat function to add 2 strings.

Q5. Article Views II 0 - 6 months: Google(2); 1 year - 2 years: LinkedIn

Problem Statement:

Write a query to find all the people who viewed **more than one** article on the **same** date.

- Save the **viewer_id** as the **id**.
- Return the result sorted by **id** in ascending order.

SQL Schema:

```
Create table If Not Exists Views (article_id int, author_id int, viewer_id int, view_date date);
Truncate table Views;
insert into Views (article_id, author_id, viewer_id, view_date) values ('1', '3', '5', '2019-08-01');
insert into Views (article_id, author_id, viewer_id, view_date) values ('3', '4', '5', '2019-08-01');
insert into Views (article_id, author_id, viewer_id, view_date) values ('1', '3', '6', '2019-08-02');
insert into Views (article_id, author_id, viewer_id, view_date) values ('2', '7', '7', '2019-08-01');
insert into Views (article_id, author_id, viewer_id, view_date) values ('2', '7', '6', '2019-08-02');
insert into Views (article_id, author_id, viewer_id, view_date) values ('4', '7', '1', '2019-07-22');
insert into Views (article_id, author_id, viewer_id, view_date) values ('3', '4', '4', '2019-07-21');
insert into Views (article_id, author_id, viewer_id, view_date) values ('3', '4', '4', '2019-07-21');
```

Sample Input:

Table: **views**

article_id	author_id	viewer_id	view_date
1	3	5	2019-08-01
3	4	5	2019-08-01
1	3	6	2019-08-02
2	7	7	2019-08-01
2	7	6	2019-08-02
4	7	1	2019-07-22
3	4	4	2019-07-21
3	4	4	2019-07-21

Sample output:

id
5
6

Sample Explanation:

- Viewer with id 5 viewed two articles on 2019-08-01.
- Viewer with id 6 viewed two articles on 2019-08-02.
- Viewer with id 4 viewed the same article two times. Hence, not included.

Q6. Banned Accounts 1 year - 2 years: Audible

Problem Statement:

Write a query to find the **account_id** of the accounts that should be banned from Leetflex.

An account should be banned if it was logged in at some moment from **two different** IP addresses.

- Return the output ordered by **account_id** in ascending order.

SQL Schema:

```
Create table If Not Exists LogInfo (account_id int, ip_address int, login datetime, logout datetime);
Truncate table LogInfo;
insert into LogInfo (account_id, ip_address, login, logout) values ('1', '1', '2021-02-01 09:00:00', '2021-02-01 09:30:00');
insert into LogInfo (account_id, ip_address, login, logout) values ('1', '2', '2021-02-01 08:00:00', '2021-02-01 11:30:00');
insert into LogInfo (account_id, ip_address, login, logout) values ('2', '6', '2021-02-01 20:30:00', '2021-02-01 22:00:00');
insert into LogInfo (account_id, ip_address, login, logout) values ('2', '7', '2021-02-02 20:30:00', '2021-02-02 22:00:00');
insert into LogInfo (account_id, ip_address, login, logout) values ('3', '9', '2021-02-01 16:00:00', '2021-02-01 16:59:59');
insert into LogInfo (account_id, ip_address, login, logout) values ('3', '13', '2021-02-01 17:00:00', '2021-02-01 17:59:59');
insert into LogInfo (account_id, ip_address, login, logout) values ('4', '10', '2021-02-01 16:00:00', '2021-02-01 17:00:00');
insert into LogInfo (account_id, ip_address, login, logout) values ('4', '11', '2021-02-01 17:00:00', '2021-02-01 17:59:59');
```

Sample Input:

Table: loginfo

account_id	ip_address	login	logout
1	1	2021-02-01 09:00:00	2021-02-01 09:30:00
1	2	2021-02-01 08:00:00	2021-02-01 11:30:00
2	6	2021-02-01 20:30:00	2021-02-01 22:00:00
2	7	2021-02-02 20:30:00	2021-02-02 22:00:00
3	9	2021-02-01 16:00:00	2021-02-01 16:59:59
3	13	2021-02-01 17:00:00	2021-02-01 17:59:59
4	10	2021-02-01 16:00:00	2021-02-01 17:00:00
4	11	2021-02-01 17:00:00	2021-02-01 17:59:59

Sample output:

account_id
1
4

Sample Explanation:

- Account ID 1 --> The account was active from "2021-02-01 09:00:00" to "2021-02-01 09:30:00" with two different IP addresses (1 and 2). It should be banned.
- Account ID 2 --> The account was active from two different addresses (6, 7) but at two different times.
- Account ID 3 --> The account was active from two different addresses (9, 13) on the same day but they do not intersect at any moment.
- Account ID 4 --> The account was active from "2021-02-01 17:00:00" to "2021-02-01 17:00:00" with two different IP addresses (10 and 11). It should be banned.

Q7. No Job history

Problem Statement:

Display all the details of the employees who did **not work** at any job in the **past**.

- Return **all** the columns from the employee's table.
- Return the result ordered by **employee_id** in ascending order.

NOTE:

- To get the details of the employee's previous jobs refer to the job_history table.
- An employee is present in the job_history table if has worked before.

Dataset Description is exactly same as Q3.

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

Table: job_history

employee_id	start_date	end_date	job_id	department_id
101	1989-09-21	1993-10-27	AC_ACCOUNT	110
101	1993-10-28	1997-03-15	AC_MGR	110
102	1993-01-13	1998-07-24	IT_PROG	60

Sample Output:

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
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

Explanation: Only employess with id 101 and 102 have a job history.