Q1. Last Month's Friendly Movies

1 year - 2 years: Amazon

SQL Schema:

```
create database friendly_movies;
use friendly_movies;
Create table If Not Exists Content (content_id int, title varchar(255), Kids_content enum('Y', 'N'), content_type varchar(255));
Truncate table Content;
insert into Content (content_id, title, Kids_content, content_type) values ('1', 'LC Movie', 'N', 'Movies');
insert into Content (content_id, title, Kids_content, content_type) values ('2', 'Alg. for Kids', 'Y', 'Series');
insert into Content (content_id, title, Kids_content, content_type) values ('3', 'Database Sols', 'N', 'Series');
insert into Content (content_id, title, Kids_content, content_type) values ('4', 'Aladdin', 'Y', 'Movies');
insert into Content (content_id, title, Kids_content, content_type) values ('5', 'Cinderella', 'Y', 'Movies');
Create table If Not Exists TVProgram (program_date datetime, content_id int, channel varchar(255));
Truncate table TVProgram;
insert into TVProgram (program_date, content_id, channel) values ('2020-06-10 08:00:00', '1', 'LC-Channel');
insert into TVProgram (program_date, content_id, channel) values ('2020-05-11 12:00:00', '2', 'LC-Channel');
insert into TVProgram (program_date, content_id, channel) values ('2020-05-12 12:00:00', '3', 'LC-Channel');
insert into TVProgram (program_date, content_id, channel) values ('2020-05-13 14:00:00', '4', 'Disney Ch');
insert into TVProgram (program_date, content_id, channel) values ('2020-06-18 14:00:00', '4', 'Disney Ch');
insert into TVProgram (program_date, content_id, channel) values ('2020-07-15 16:00:00', '5', 'Disney Ch');
```

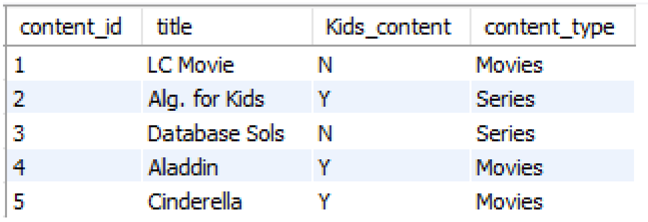
Problem Statement:

Write a SQL query to report the **unique** titles of the **kid-friendly** movies streamed in **June 2020**.

- Use the given **Content** and **TVProgram** tables.
- Return the result table ordered by **title** in ascending order.

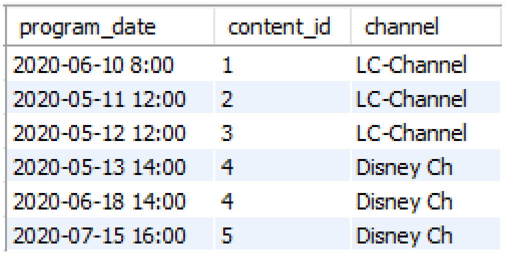
Sample Input:

Table: Content



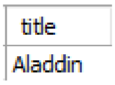
content_id	title	Kids_content	content_type
1	LC Movie	N	Movies
2	Alg. for Kids	Y	Series
3	Database Sols	N	Series
4	Aladdin	Y	Movies
5	Cinderella	Y	Movies

Table: TVProgram



program_date	content_id	channel
2020-06-10 8:00	1	LC-Channel
2020-05-11 12:00	2	LC-Channel
2020-05-12 12:00	3	LC-Channel
2020-05-13 14:00	4	Disney Ch
2020-06-18 14:00	4	Disney Ch
2020-07-15 16:00	5	Disney Ch

Sample Output:



title
Aladdin

Explanation:

- "LC Movie" is not a content for kids.
- "Alg. for Kids" is not a movie.
- "Database Sols" is not a movie
- "Alladin" is a movie, content for kids and was streamed in June 2020.
- "Cinderella" was not streamed in June 2020.

Q2. Special Bonus

6 months - 1 year: Apple-2

SQL Schema:

```
Create table If Not Exists Employees (employee_id int, name varchar(30), salary int);
Truncate table Employees;
insert into Employees (employee_id, name, salary) values ('2', 'Meir', '3000');
insert into Employees (employee_id, name, salary) values ('3', 'Michael', '3800');
insert into Employees (employee_id, name, salary) values ('7', 'Addilyn', '7400');
insert into Employees (employee_id, name, salary) values ('8', 'Juan', '6100');
insert into Employees (employee_id, name, salary) values ('9', 'Kannon', '7700');
```

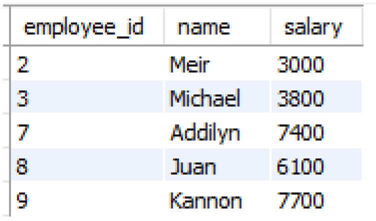
Problem Statement:

Write a query to calculate the bonus of each employee. The bonus of an employee is `100%` of their salary if the ID of the employee is an **odd number** and the employee name does not start with the character **'M'**. The bonus of an employee is `0` otherwise.

- Return the result table ordered by `employee_id` in ascending manner.

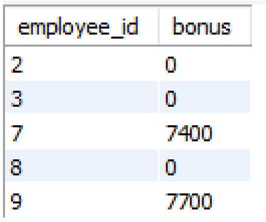
Sample Input:

Table: `employees`



employee_id	name	salary
2	Meir	3000
3	Michael	3800
7	Addilyn	7400
8	Juan	6100
9	Kannon	7700

Sample output:



employee_id	bonus
2	0
3	0
7	7400
8	0
9	7700

Explanation:

- The employees with IDs 2 and 8 get 0 bonus because they have an even employee_id.
- The employee with ID 3 gets 0 bonus because their name starts with 'M'.
- The rest of the employees get a 100% bonus.

Q3. Judgement of Triangle 1 year - 2 years: Facebook (2)

SQL Schema:

```
Create table If Not Exists Triangle (x int, y int, z int);
Truncate table Triangle;
insert into Triangle (x, y, z) values ('13', '15', '30');
insert into Triangle (x, y, z) values ('10', '20', '15');
```

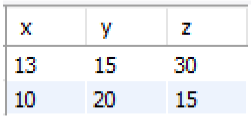
Problem Statement:

Write a SQL query to report whether each triad of the three line segments in the given data can form a triangle or not.

Note: Return the result table ordered by `x`, `y` and `z` in ascending order.

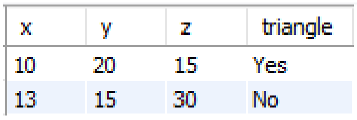
Sample Input:

Table: `triangle`



x	y	z
13	15	30
10	20	15

Sample output:



x	y	z	triangle
10	20	15	Yes
13	15	30	No

Explanation:

Three line segments can form a triangle only if the sum of any of the two segments is larger than the third one.

- x = 13, y = 15, z = 30
  - 13 + 15 (28) > 30 (No)
  - 15 + 30 (45) > 13 (Yes)
  - 13 + 30 (43) > 15 (Yes)
  - Since all three conditions are not satisfied these three segments can't form a triangle.
- x = 10, y = 20, z = 15
  - 10 + 20 (30) > 15 (Yes)
  - 20 + 15 (35) > 10 (Yes)
  - 10 + 15 (25) > 20 (Yes)
  - Since all three conditions are satisfied these three segments can form a triangle.

Q4. Job like sales

Problem Statement:

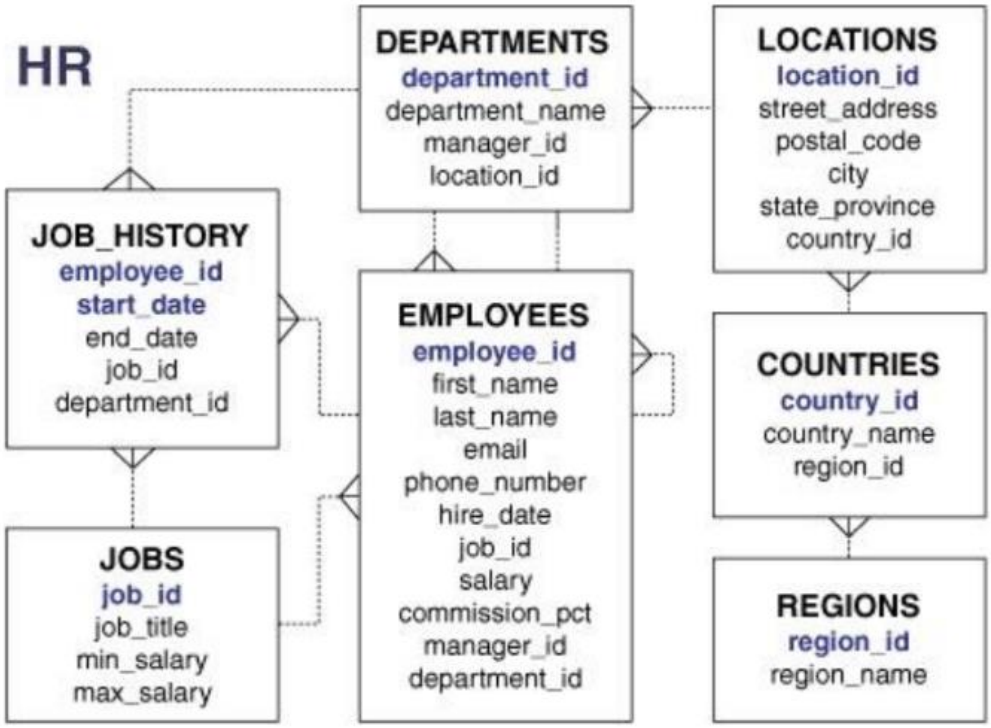
Display the details of the employees who **had** job titles like '`sales`' in the **past** and the **min_salary** is greater than or equal to **6000**.

- Return the columns '**employee_id**', '**department_name**', '**job_id**', '**job_title**', and '**min_salary**'.
- Return the employee's current information for the columns 'employee_id', and 'department_name'.
- Return the employee's past information for the columns 'job_id', 'job_title', and 'min_salary'.
- Return the output ordered by **employee_id** and **min_salary** in ascending order.

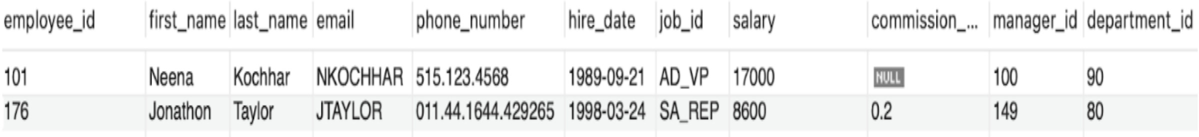
NOTE:

1. To get the **min_salary** refer to the jobs table.
2. Refer to the **job_history** table to get the details of past jobs.
3. An employee might have worked in multiple jobs in the past whose record will be available in job_history.
4. If any employee hasn't worked in any jobs in the past, his record wouldn't be present in the job_history table.

Dataset Description:

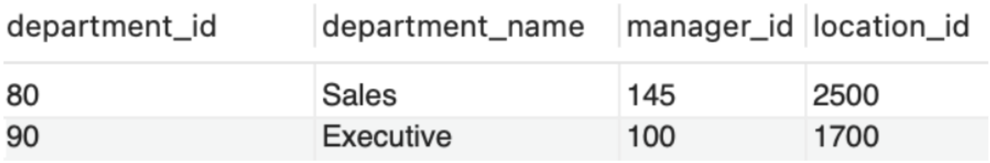


Sample Input:
Table: employees



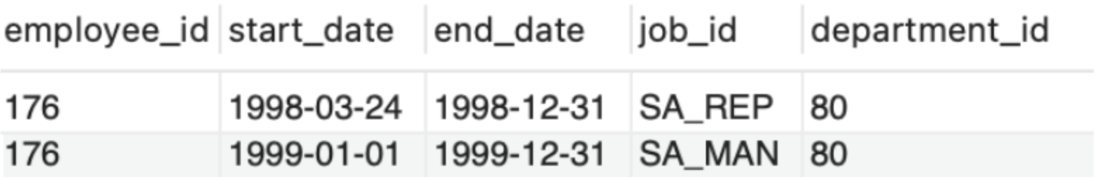
employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_...	manager_id	department_id
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1989-09-21	AD_VP	17000	NULL	100	90
176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	1998-03-24	SA_REP	8600	0.2	149	80

Table: departments



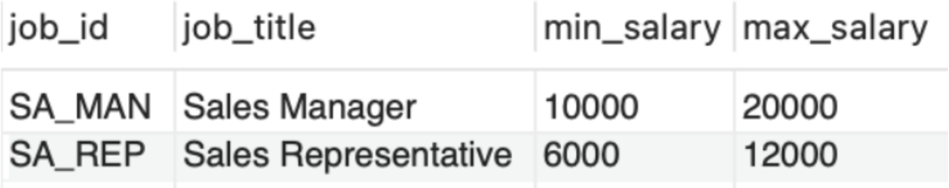
department_id	department_name	manager_id	location_id
80	Sales	145	2500
90	Executive	100	1700

Table: job_history



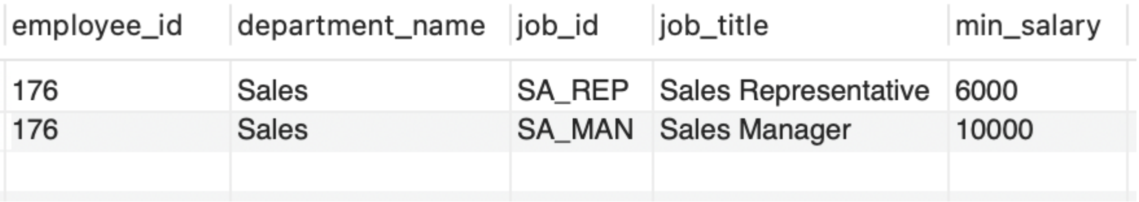
employee_id	start_date	end_date	job_id	department_id
176	1998-03-24	1998-12-31	SA_REP	80
176	1999-01-01	1999-12-31	SA_MAN	80

Table: jobs



job_id	job_title	min_salary	max_salary
SA_MAN	Sales Manager	10000	20000
SA_REP	Sales Representative	6000	12000

Sample Output:



employee_id	department_name	job_id	job_title	min_salary
176	Sales	SA_REP	Sales Representative	6000
176	Sales	SA_MAN	Sales Manager	10000

Q5. Salary Bins

Problem Statement:

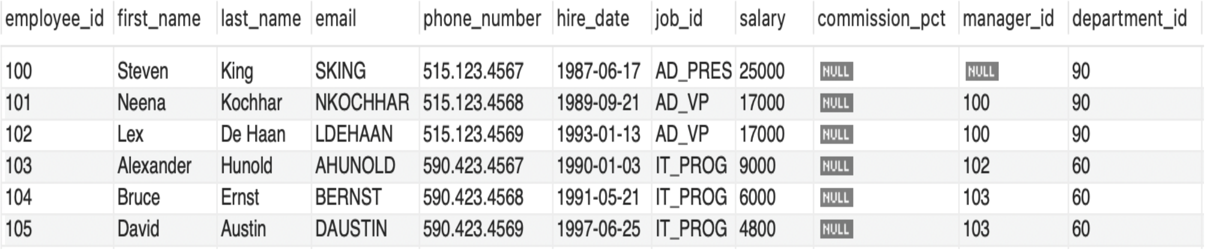
Based on the employee's salary, divide the employees into three different classes.

1. Salary **greater than** 20,000 (i.e, excluding 20,000) as '**Class A**'
2. Salary **between** 10,000 to 20,000 (i.e, including both 10,000 and 20,000) as '**Class B**'
3. Salary **less than** 10,000 (i.e, excluding 10,000) as '**Class C**'. Return the new column as 'Salary_bin'.

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

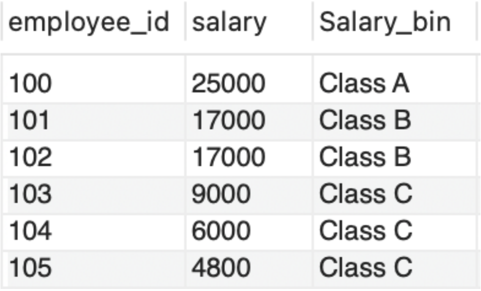
Dataset Description is the same as Q4.

Sample Input:



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	salary	Salary_bin
100	25000	Class A
101	17000	Class B
102	17000	Class B
103	9000	Class C
104	6000	Class C
105	4800	Class C

Q6. Accountant

Problem Statement:

Using the **employees** table, create a new column as '**Accountant**'.

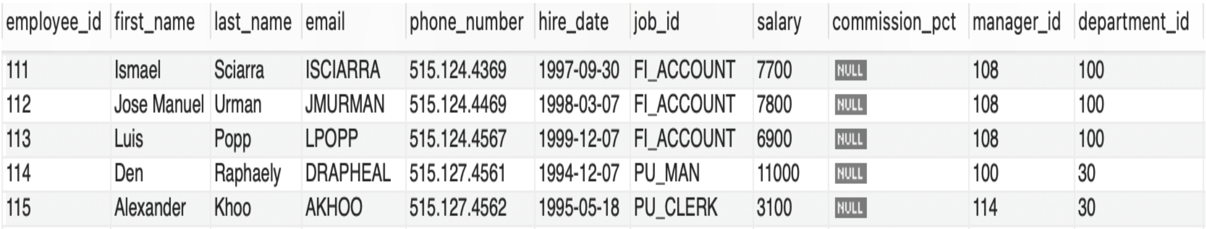
If the employees are working at the '**FI_ACCOUNT**' or '**AC_ACCOUNT**' designation then label it as 1, else label all other designations as 0.

- Return the columns '**employee_id**', '**first_name**', '**last_name**', '**salary**', '**Accountant**'.
- Return the output ordered by **employee_id** in ascending order.

Dataset Description is the same as Q4.

Sample Input:

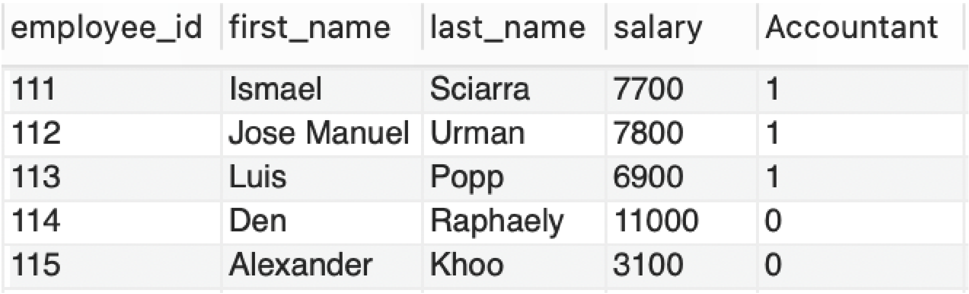
Table: employees



employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id	department_id
111	Ismael	Sciarra	ISCIARRA	515.124.4369	1997-09-30	FI_ACCOUNT	7700	NULL	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	1998-03-07	FI_ACCOUNT	7800	NULL	108	100
113	Luis	Popp	LPOPP	515.124.4567	1999-12-07	FI_ACCOUNT	6900	NULL	108	100
114	Den	Raphaely	DRAPHEAL	515.127.4561	1994-12-07	PU_MAN	11000	NULL	100	30
115	Alexander	Khoo	AKHOO	515.127.4562	1995-05-18	PU_CLERK	3100	NULL	114	30

- Refer to the column **job_id** to get the details of the designation.

Sample Output:



employee_id	first_name	last_name	salary	Accountant
111	Ismael	Sciarra	7700	1
112	Jose Manuel	Urman	7800	1
113	Luis	Popp	6900	1
114	Den	Raphaely	11000	0
115	Alexander	Khoo	3100	0