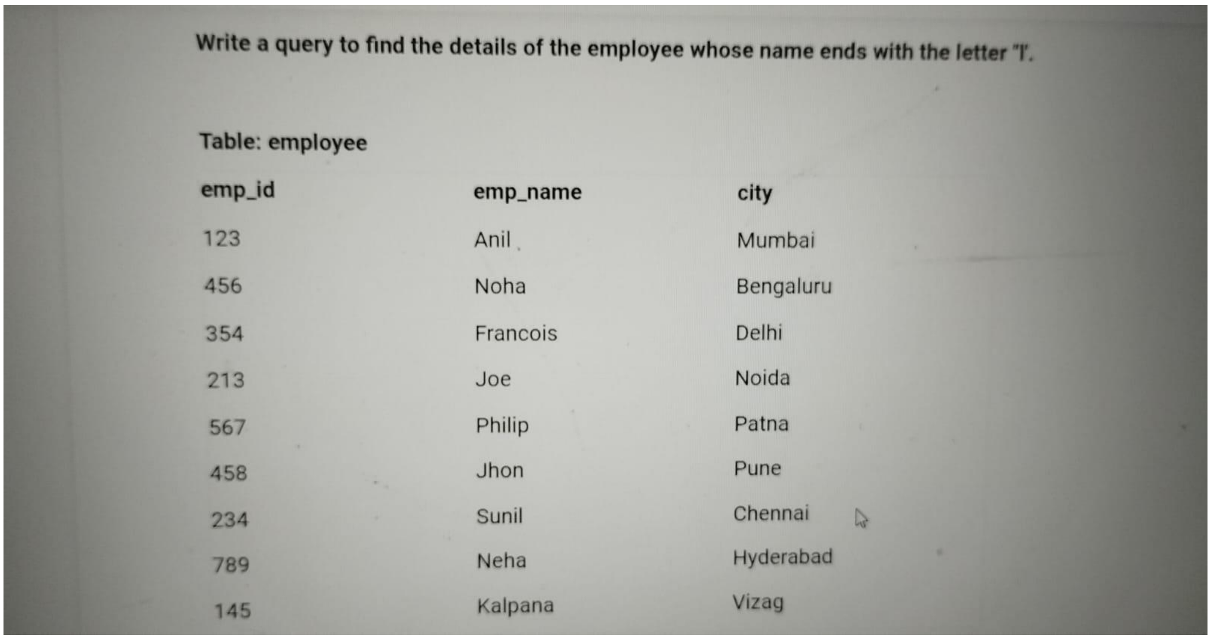
E



Write a query to find the details of the employee whose name ends with the letter 'I'.

Table: employee

emp_id	emp_name	city
123	Anil	Mumbai
456	Noha	Bengaluru
354	Francois	Delhi
213	Joe	Noida
567	Philip	Patna
458	Jhon	Pune
234	Sunil	Chennai
789	Neha	Hyderabad
145	Kalpana	Vizag

-- Query to find employees whose name ends with 'T'

SELECT *

FROM employee

WHERE emp_name LIKE '%T';

-- Query to perform top 6 operations based on employee ID

SELECT *

FROM employee

ORDER BY emp_id

LIMIT 6;

1. **Count the total number of employees:**

```sql
SELECT COUNT(*) AS total_employees
FROM employee;
```

2. **Find the maximum employee ID:**

```sql
SELECT MAX(emp_id) AS max_emp_id
FROM employee;
```

3. **Get the employees sorted by city in ascending order:**

```sql
SELECT *
FROM employee
ORDER BY city ASC;
```

4. **Find employees whose name starts with 'A':**

```sql
SELECT *
FROM employee
WHERE emp_name LIKE 'A%';
```

5. **Find the average employee ID:**

```sql
SELECT AVG(emp_id) AS avg_emp_id
FROM employee;

*E2*
```

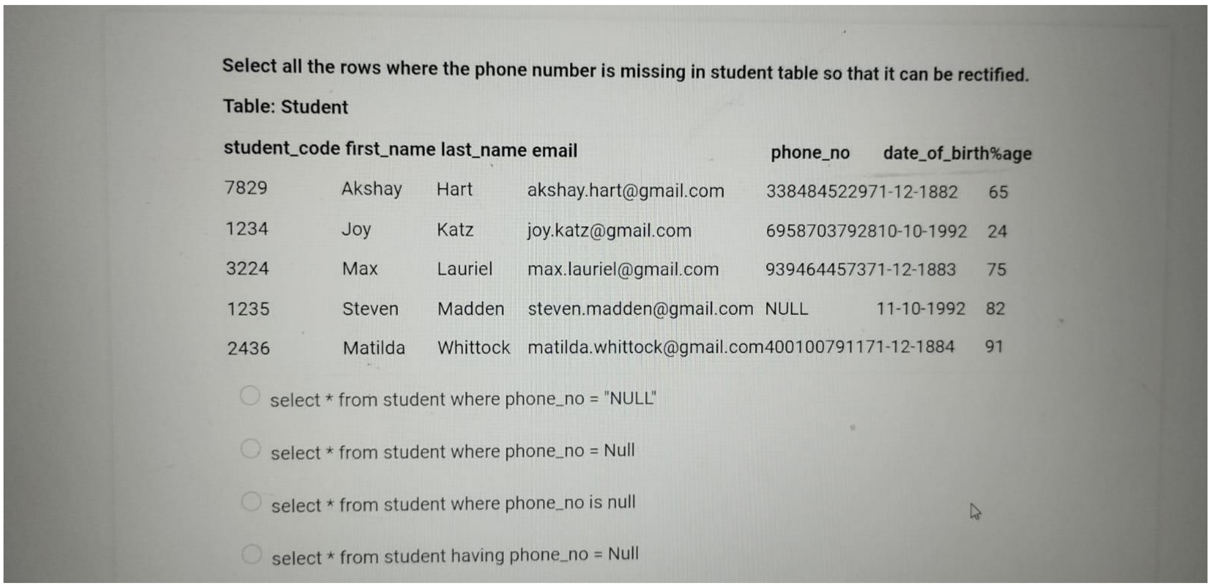

Select all the rows where the phone number is missing in student table so that it can be rectified.

Table: Student

student_code	first_name	last_name	email	phone_no	date_of_birth	%age
7829	Akshay	Hart	akshay.hart@gmail.com	3384845229	71-12-1882	65
1234	Joy	Katz	joy.katz@gmail.com	6958703792	810-10-1992	24
3224	Max	Lauriel	max.lauriel@gmail.com	9394644573	71-12-1883	75
1235	Steven	Madden	steven.madden@gmail.com	NULL	11-10-1992	82
2436	Matilda	Whittock	matilda.whittock@gmail.com	4001007911	71-12-1884	91

○ select * from student where phone_no = "NULL"

○ select * from student where phone_no = Null

○ select * from student where phone_no is null

○ select * from student having phone_no = Null

Query to Select Rows Where Phone Number is Missing:

```sql

SELECT *

FROM Student

WHERE phone_no IS NULL;

```

5 Different Operations with this Data:

1. **Calculate the Average Age:**

- Assuming the current date is needed to calculate age, you can calculate the average age of the students.

```sql

SELECT AVG(DATEDIFF(CURDATE(), date_of_birth) / 365.25) AS average_age

FROM Student;

```

2. **Count the Total Number of Students:**

```sql
SELECT COUNT(*) AS total_students
FROM Student;
```

3. **Find Students Whose Percentage is Above 80%:**

```sql
SELECT *
FROM Student
WHERE percentage > 80;
```

4. **Get the Students Sorted by Last Name:**

```sql
SELECT *
FROM Student
ORDER BY last_name ASC;
```

5. **Find the Student with the Earliest Date of Birth:**

```sql
SELECT *
FROM Student
ORDER BY date_of_birth ASC
LIMIT 1;
```

E3

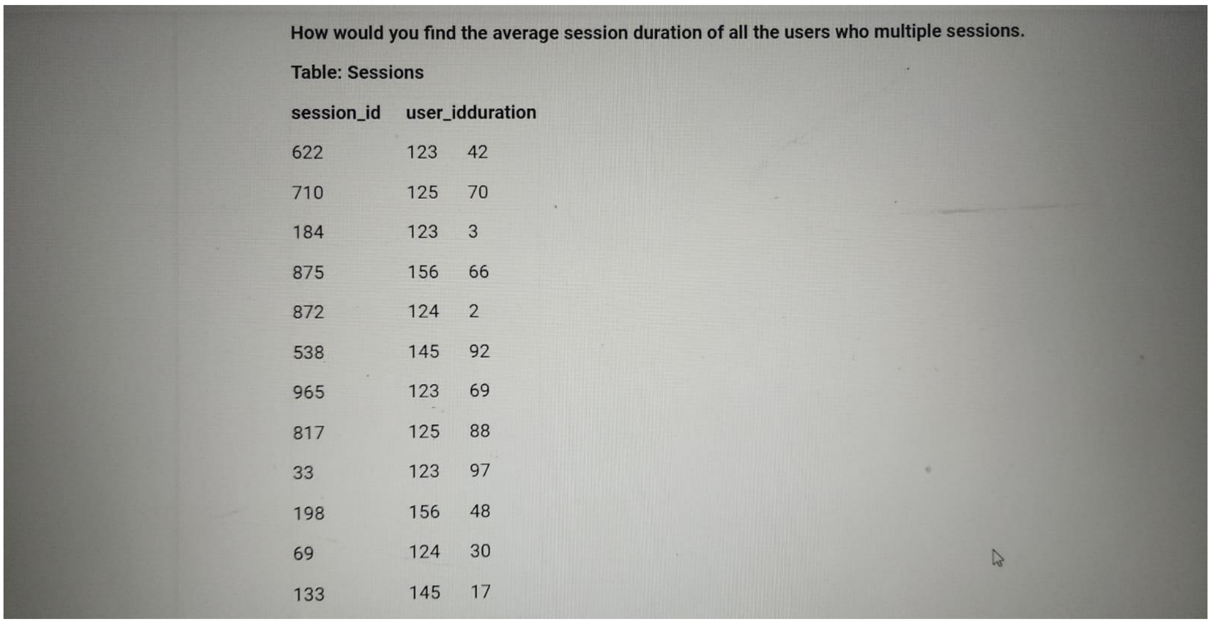

How would you find the average session duration of all the users who multiple sessions.

Table: Sessions

session_id	user_id	duration
622	123	42
710	125	70
184	123	3
875	156	66
872	124	2
538	145	92
965	123	69
817	125	88
33	123	97
198	156	48
69	124	30
133	145	17

1. **Identify users who have multiple sessions.**

2. **Calculate the average session duration for those users.**

Here's how you can do it in SQL:

```sql
WITH multiple_sessions AS (
   -- Step 1: Find users with multiple sessions
   SELECT user_id
   FROM Sessions
   GROUP BY user_id
   HAVING COUNT(session_id) > 1
)

-- Step 2: Calculate the average session duration for those users
SELECT AVG(duration) AS average_duration
FROM Sessions
```

WHERE user_id IN (SELECT user_id FROM multiple_sessions);

```
```

Explanation:

- Step 1: The `WITH multiple_sessions` clause creates a temporary result set that includes only the `user_id` of users who have more than one session.

- Step 2: The main query then selects the average duration of sessions for these users. The `IN` clause ensures that only sessions of users with multiple sessions are considered.

E4

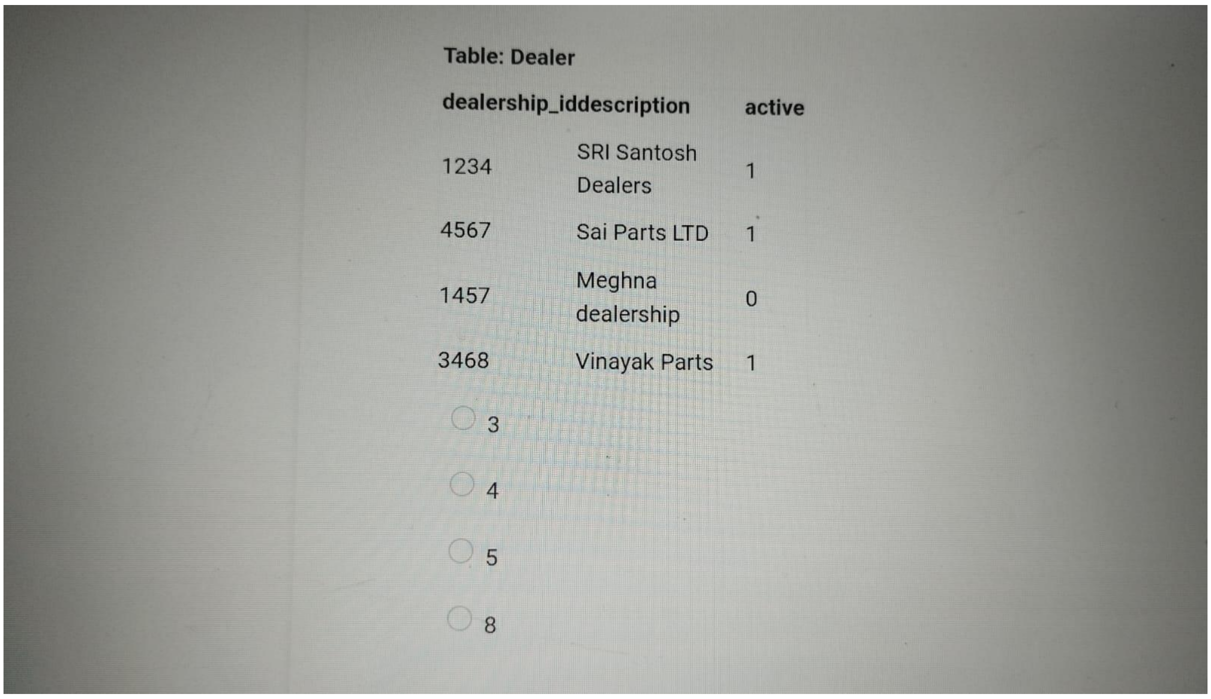


Table: Dealer

dealership_id	description	active
1234	SRI Santosh Dealers	1
4567	Sai Parts LTD	1
1457	Meghna dealership	0
3468	Vinayak Parts	1

- ◯ 3
- ◯ 4
- ◯ 5
- ◯ 8

Dealer Table

dealership_id	description	active
1	SRI Santosh	1
1234	Dealers	0
4567	Sai Parts LTD	1
1457	Meghna	1

3468	dealership	0
3	Vinayak Parts	1
04	(Missing Data)	0
5	(Missing Data)	8

SQL Operations on the Dealer Table:

1. **Select all active dealerships:**

   ```sql
   SELECT *
   FROM Dealer
   WHERE active = 1;
   ```

2. **Count the number of active dealerships:**

   ```sql
   SELECT COUNT(*) AS active_dealerships
   FROM Dealer
   WHERE active = 1;
   ```

3. **Find dealerships with incomplete descriptions (assuming missing or NULL descriptions are incomplete):**

   ```sql
   SELECT *
   FROM Dealer
   WHERE description IS NULL OR description = '';
   ```

4. **List all dealerships sorted by dealership ID:**

   ```sql
   ```

SELECT *

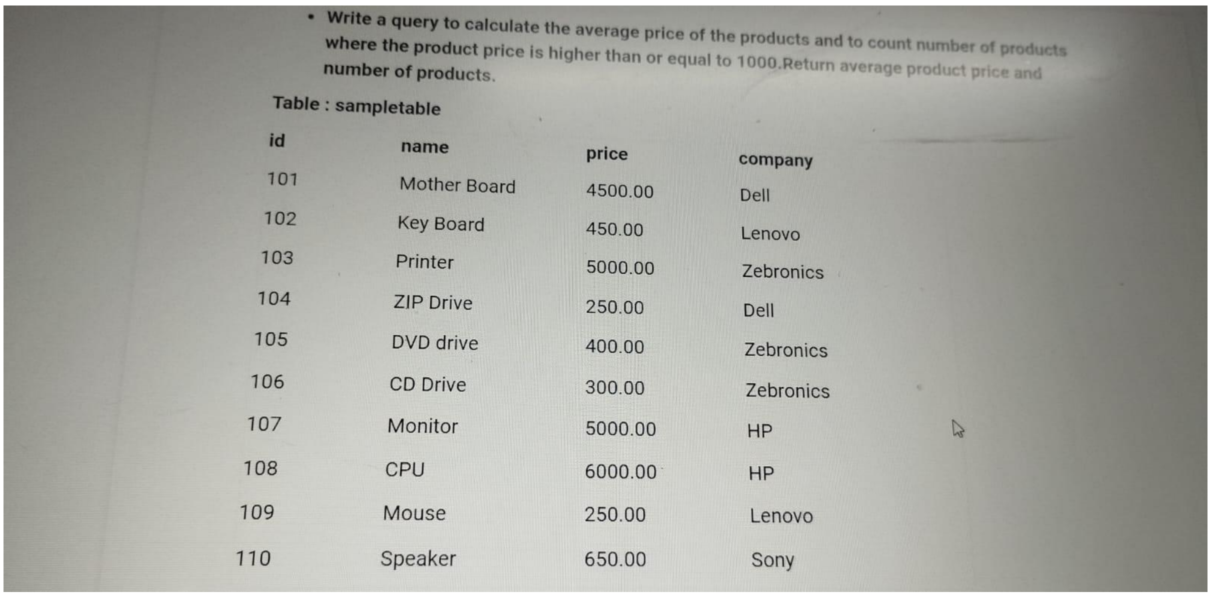
FROM Dealer

ORDER BY dealership_id ASC;

5. **Update the `active` status of a specific dealership (for example, set `active` to `0` for `dealership_id = 1234`):**

```sql

UPDATE Dealer

SET active = 0

WHERE dealership_id = 1234;

```

E5



- Write a query to calculate the average price of the products and to count number of products where the product price is higher than or equal to 1000.Return average product price and number of products.

Table : sampletable

id	name	price	company
101	Mother Board	4500.00	Dell
102	Key Board	450.00	Lenovo
103	Printer	5000.00	Zebronics
104	ZIP Drive	250.00	Dell
105	DVD drive	400.00	Zebronics
106	CD Drive	300.00	Zebronics
107	Monitor	5000.00	HP
108	CPU	6000.00	HP
109	Mouse	250.00	Lenovo
110	Speaker	650.00	Sony

```sql

SELECT

  AVG(price) AS average_price,

  COUNT(*) AS number_of_products_above_1000

FROM
```

```
    sampletable
WHERE
    price >= 1000;
```

Explanation:

- **AVG(price) AS average_price**: This calculates the average price of all products.
- **COUNT(*) AS number_of_products_above_1000**: This counts the number of products where the price is higher than or equal to 1000.
- **WHERE price >= 1000**: This condition filters the products to include only those with a price greater than or equal to 1000.

This query will return two results:

1. The average price of all products.
2. The count of products where the price is 1000 or higher.