121/200XP 15 15

Calculating Free-to-Paid Conversion Rate with SQL Project

Calculating the Fraction of Students Who Convert to Paying Ones after Starting a Course

Free

Beginner

With Hristina Hristova

Type: **Practice project**

Duration: **2 Hours**

Status: **Done**

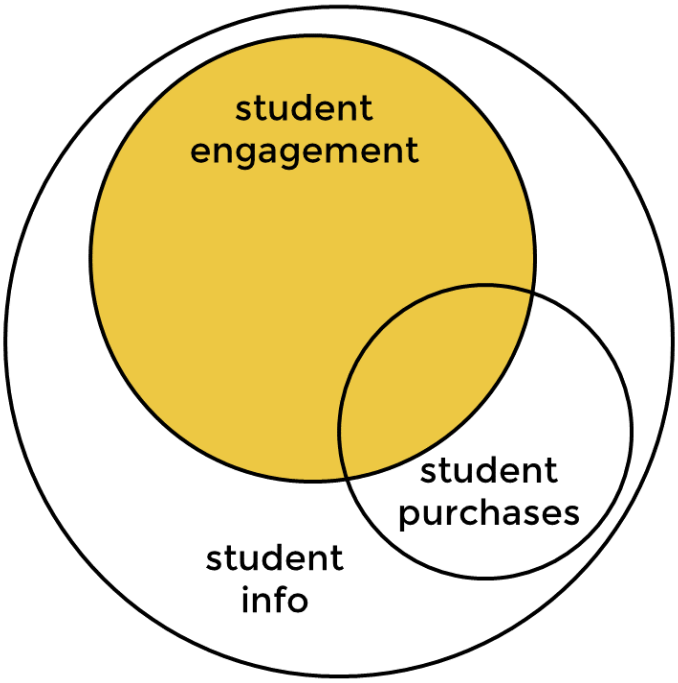
View Instructions All solution files

Description Solution

Part 1: Create the Subquery

1. First, remember to import the `db_course_conversions` database and refresh the Schemas pane to see it appear. Apply the `USE` keyword to use the named database as the default (current) one.

```
USE db_course_conversions;
```



Let's first join `student_engagement` with the `student_info` table. The Venn diagram hints that all students in the engagement table are also present in the info table. Therefore, upon joining these two tables, we'll retrieve all students from the engagement table with their registration and engagement dates, which will be essential in the next step. We should state which field we are joining these tables on, the `student_id` field.

```
FROM
    student_engagement e
        JOIN
    student_info i ON e.student_id = i.student_id
        ???
    student_purchases p ON ???
```

Next, we need to join this resulting set with the `student_purchases` table to exclude all students who haven't watched a lecture. This can be achieved with the `LEFT JOIN` clause, again joining the table on the `student_id` field.

```
FROM
    student_engagement e
        JOIN
    student_info i ON e.student_id = i.student_id
```

3. Now select the fields asked for in the task. The first is the `student_id` column, which we can extract from the `student_engagement` table (aliased `e`). We chose the engagement table because it stores all the records we want to obtain. (Refer to the shaded region in the Venn diagram.)

```
SELECT
    e.student_id,
FROM
    ...
```

We should select a student's registration date in the `student_info` table (aliased `i`).

```
SELECT
    e.student_id,
    i.date_registered,
FROM
    ...
```

Then, we must retrieve the first-time engagement and purchase dates from the engagement and purchase tables, respectively. We can achieve this by employing the `MIN` aggregate function. When applied to numbers, this function returns the smallest number in a set. It analogously retrieves the earliest date in a set when used on dates.

```
SELECT
    e.student_id,
    i.date_registered,
    MIN(e.date_watched) AS first_date_watched,
    MIN(p.date_purchased) AS first_date_purchased,
FROM
    ...
```

Finally, as the task hint suggests, we can use the `DATEDIFF` function to find the day difference between the two dates. Note that the date that comes later should be placed as a first argument and the one that comes earlier as

```
SELECT
    e.student_id,
    i.date_registered,
    MIN(e.date_watched) AS first_date_watched,
    MIN(p.date_purchased) AS first_date_purchased,
    DATEDIFF(MIN(e.date_watched), i.date_registered) AS days_diff_re
    DATEDIFF(MIN(p.date_purchased),
            MIN(e.date_watched)) AS days_diff_watch_purch
FROM
    ...
```

4. In the previous step, we used the `MIN` aggregate function to find the earliest engagement and purchase dates. Note, however, that to find the earliest days *per student*, we need to group by the `student_id` field from the engagement table (aliased `e`).

```
SELECT
    ...
FROM
    ...
GROUP BY e.student_id;
```

5. Finally, for this task, we should filter the data so that the earliest engagement date comes before or is on the same day as the earliest purchase date. We can achieve this using the following condition:

```
first_date_watched <= first_date_purchased
```

   We also need to include all records whose `first_date_purchased` column equals `NULL`—indicative that the student hasn't made a purchase.

```
first_date_purchased IS NULL
```

aggregation in the third and fourth steps.

```
SELECT
    ...
FROM
    ...
GROUP BY
    …
HAVING first_date_purchased IS NULL
    OR first_date_watched <= first_date_purchased;
```

Part 2: Create the Main Query

1. Surround the subquery you previously created in parentheses and give it an alias, say `a`.

```
(SELECT
    ...
FROM
    ...
GROUP BY
    ...
HAVING ...) a;
```

2. One way to calculate the first of the three metrics (`conversion_rate`) is to count the number of occurrences in the `first_date_purchased` column and divide the result by the number of occurrences in the `first_date_watched` column. The `COUNT` function will not account for the `NULL` values in the former column, giving the number of students who purchased a subscription after watching a lecture. We round the number to two decimal places and multiply it by 100 to retrieve the result in percentages.

```
            2) * 100 AS conversion_rateFROM
    (...) a;
```

You could use other column combinations to obtain the same result.

3. The second metric, `av_reg_watch`, can be calculated by summing all records from the `days_diff_reg_watch` column and dividing the result by the number of records in the same column. This will give the average duration between the date of registration and the date of first-time engagement.

```
SELECT
    ROUND(COUNT(first_date_purchased) / COUNT(first_date_watched),
            2) * 100 AS conversion_rate,
    ROUND(SUM(days_diff_reg_watch) / COUNT(days_diff_reg_watch),
            2) AS av_reg_watch,
FROM
    (...) a;
```

4. Finally, the third metric is analogously found by summing all records from the `days_diff_watch_purch` column and dividing the result by the number of records in this column. This will give the average duration between first-time engagement and first-time purchase dates.

```
SELECT
    ROUND(COUNT(first_date_purchased) / COUNT(first_date_watched),
            2) * 100 AS conversion_rate,
    ROUND(SUM(days_diff_reg_watch) / COUNT(days_diff_reg_watch),
```

Part 3: Interpretation

when more involved data analysis tasks are required. With that in mind, below, we try to make sense of the three metrics we've retrieved.

1. **Conversion Rate**

   Let's first discuss the result we obtained for the free-to-paid conversion rate metric. The fraction of students who purchase monthly, quarterly, or annual subscriptions from those who watch a lecture is about 11%—i.e., for every 100 students who come to the 365 platform, roughly 11 of them purchase a subscription. At first glance, this number seems relatively low, but let's dig a bit deeper.

   A significant number of students register on the platform out of curiosity. Nevertheless, we can outline why most students aren't prompted to benefit from the program entirely. One factor contributing to this could be that we're targeting a broader audience rather than focusing specifically on data science enthusiasts eager to begin their journey in the field.

   Second, since our platform targets a beginner audience, students may need clarification about what to start with. Should they first invest weeks in mastering an object-oriented programming language such as Python, a query language such as SQL, or maybe a data visualization software like Tableau? What prerequisites are necessary for each of these tools? In August 2023, the team at 365 put effort into making its audience's journey much easier by introducing an onboarding sequence that creates a customized learning path for each of its students. This way, users will know exactly where to start and how to continue.

   Still, some users might need more time to embark on a data science journey. They might be college students whose exam periods have just started or working people who can't dedicate the desired time.

   Finally, we must consider that some users might not fancy the platform and would instead take the first steps toward data science elsewhere. Still, whatever the reason, reaching out to customers is essential, pinpointing any flaws and striving towards a better product.

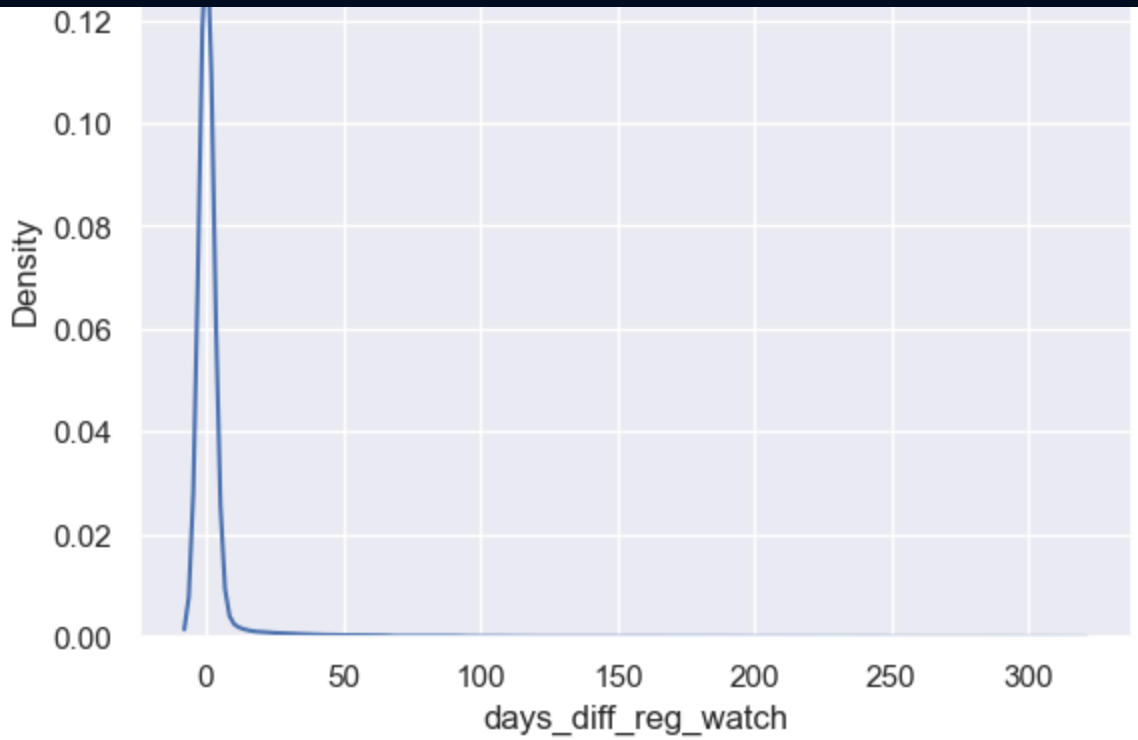
2. **Average Duration Between Registration and First-Time Engagement**

registering on the platform. Ideally, it would be great if a new student started watching a lecture on their first day. Every other element the platform offers (exams, projects, career tracks) requires more attention, while the lessons are easy to check out. It's worth diving a bit deeper into this analysis.

The dataset's average is a metric that shouldn't be studied in a vacuum because outliers heavily affect it. It's, therefore, essential to study the other two metrics that typically come hand-in-hand with the average: median and mode. The median tells us which number sits in the middle of a dataset —assuming it's ordered—while the mode is the number that occurs most often in a dataset. Calculating these is not as straightforward in SQL—so you can use another tool. I've chosen Python. The results are as follows:

- Mode: 0
- Median: 0
- Mean: 3.42

The number that repeats the most in the data is 0. Additionally, the number that sits in the middle of the dataset is also 0. Such metric values indicate the right-skewness of the data—i.e., we can find outliers to the right, towards higher values, of the data distribution. This implies that some students in the dataset have registered on the 365 platform but started watching a lecture much later. To convince ourselves, let's also study visually the distribution of the numbers.



Almost all students watch a lecture immediately after registering. Very few return to the platform to start a course several days or even a year after registration. One reason for returning could be because of a marketing campaign, a free-day campaign, etc.

3. **Average Duration Between First-Time Engagement and First-Time Purchase**

   Let's study analogously the average duration between the first-time engagement and purchase. The results we retrieved from our SQL analysis show that, on average, it takes students roughly 24 days to purchase a