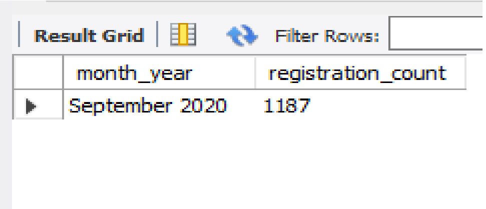
Technical Assessment - Greg Sikorsky - FanDuel - 05/21/25

1. Given the above data sets show what month and year (Eg. March 2019, April 2018, etc.) has the largest number of registrations? Please perform this using SQL and provide both the output and your query to produce your results.

ANSWER:



month_year	registration_count
September 2020	1187

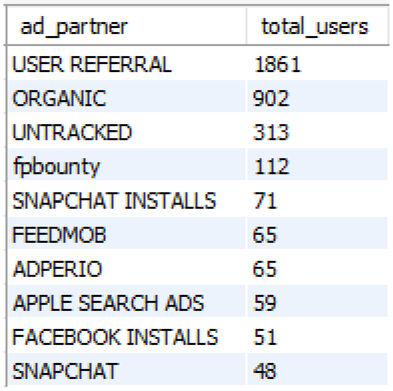
September 2020 had 1187 registrations

MySQL Code:

```
SELECT
   DATE_FORMAT(registration_date, '%M %Y') AS month_year,
   COUNT(*) AS registration_count
FROM
   user
GROUP BY
   DATE_FORMAT(registration_date, '%M %Y'),
   YEAR(registration_date),
   MONTH(registration_date)
ORDER BY
   registration_count DESC
LIMIT 1;
```

2. Given the above data sets show the top 10 ad partners by total users per First Paid Entry Credit (column “first_paid_entry_paid_media_partner_name”). Please perform this using SQL and provide both the output and your query to produce your results. Be sure to not include users who have not made a first paid entry.

ANSWER:



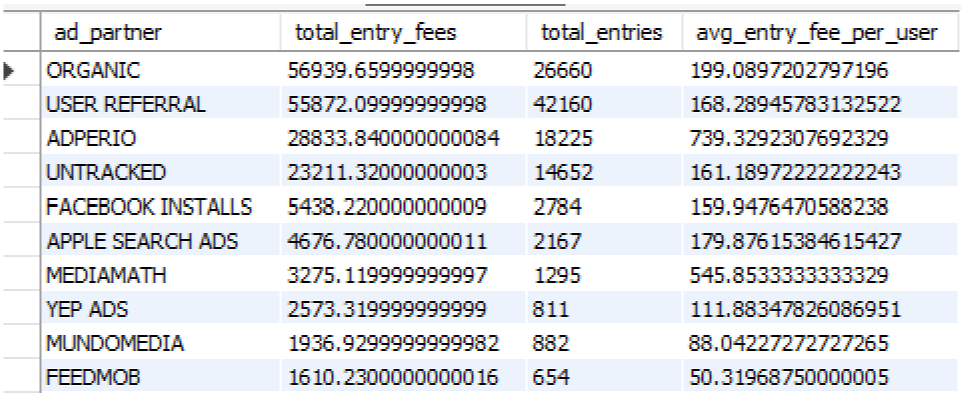
ad_partner	total_users
USER REFERRAL	1861
ORGANIC	902
UNTRACKED	313
fpbounty	112
SNAPCHAT INSTALLS	71
FEEDMOB	65
ADPERIO	65
APPLE SEARCH ADS	59
FACEBOOK INSTALLS	51
SNAPCHAT	48

MySQL Code:

```
SELECT
   first_paid_entry_paid_media_partner_name AS ad_partner,
   COUNT(*) AS total_users
FROM
   user
WHERE
   first_paid_entry_date is NOT NULL
   and first_paid_entry_paid_media_partner_name !=''
GROUP BY
   first_paid_entry_paid_media_partner_name
ORDER BY
   total_users DESC
LIMIT 10;
```

3. Like the above show the top 10 Ad Partners, grouped by First Paid Entry Credit (column “first_paid_entry_paid_media_partner_name”) and ranked by total entry fees for users who first paid entry data was in September. In addition to total entry fees, include the total count of entries and the average entry fee amount per user per ad partner. Please perform this using SQL and provide both the output and your query to produce your results.

Answer:



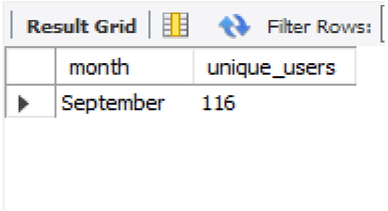
ad_partner	total_entry_fees	total_entries	avg_entry_fee_per_user
ORGANIC	56939.6599999998	26660	199.0897202797196
USER REFERRAL	55872.09999999998	42160	168.28945783132522
ADPERIO	28833.840000000084	18225	739.3292307692329
UNTRACKED	23211.32000000003	14652	161.18972222222243
FACEBOOK INSTALLS	5438.220000000009	2784	159.9476470588238
APPLE SEARCH ADS	4676.780000000011	2167	179.87615384615427
MEDIAMATH	3275.119999999997	1295	545.8533333333329
YEP ADS	2573.319999999999	811	111.88347826086951
MUNDOMEDIA	1936.9299999999982	882	88.04227272727265
FEEDMOB	1610.2300000000016	654	50.31968750000005

MySQL Code:

```
SELECT
   u.first_paid_entry_paid_media_partner_name AS ad_partner,
   SUM(e.weekly_entry_fee) AS total_entry_fees,
   SUM(e.total_entries) AS total_entries,
   SUM(e.weekly_entry_fee) / COUNT(DISTINCT u.product_account_id) AS
avg_entry_fee_per_user
FROM
   user u
JOIN
   entry e ON u.product_account_id = e.product_account_id
WHERE
   u.first_paid_entry_paid_media_partner_name IS NOT NULL
   AND MONTH(u.first_paid_entry_date) = 9
GROUP BY
   u.first_paid_entry_paid_media_partner_name
ORDER BY
   total_entry_fees DESC
LIMIT 10;
```

4. For users who registered in 2019, which month, based on a user's first_paid_entry_date, saw the largest number of unique users make an NBA entry at any point in time? Please perform this using SQL and provide both the output and your query to produce your results.

Answer:



month	unique_users
September	116

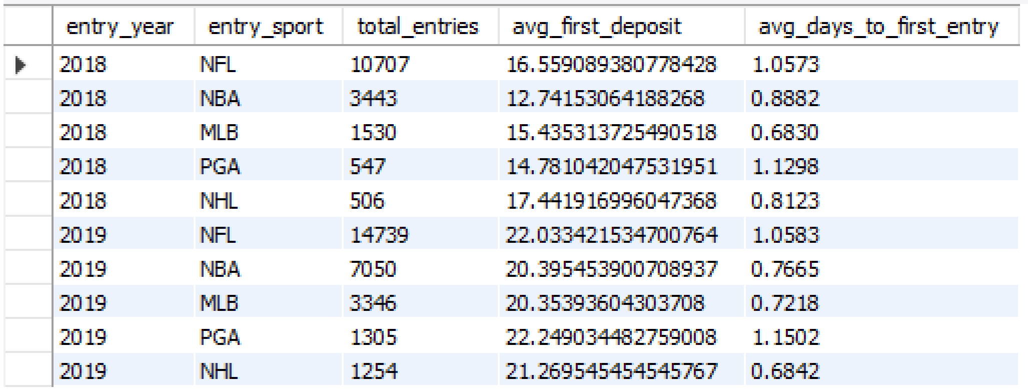
(Pre-Season? 🙂)

MySQL code:

```
SELECT
   DATE_FORMAT(u.first_paid_entry_date, '%M') AS month,
   COUNT(DISTINCT u.product_account_id) AS unique_users
FROM
   user u
JOIN
   entry e ON u.product_account_id = e.product_account_id
WHERe
   YEAR(u.registration_date) = 2019
   AND e.entry_sport = 'NBA'
   AND MONTH(u.first_paid_entry_date) IS NOT NULL
GROUP By
   MONTH(u.first_paid_entry_date),
   DATE_FORMAT(u.first_paid_entry_date, '%M')
order by
   unique_users DESC
LIMIT 1;
```

5. What were the top 5 sports in both 2018 and 2019 based on total entries for users that made their first paid entry within one week of registering? Include the average first_deposits_amount, and average time between registration and first paid entry per sport and year as well. Please write this as a single query using the “rank()” function. Please perform this using SQL and provide both the output and your query to produce your results. Note, we want to know the top 5 sports for each year not overall in 2018 and 2019 combined.

ANSWER:



entry_year	entry_sport	total_entries	avg_first_deposit	avg_days_to_first_entry
2018	NFL	10707	16.559089380778428	1.0573
2018	NBA	3443	12.74153064188268	0.8882
2018	MLB	1530	15.435313725490518	0.6830
2018	PGA	547	14.781042047531951	1.1298
2018	NHL	506	17.441916996047368	0.8123
2019	NFL	14739	22.033421534700764	1.0583
2019	NBA	7050	20.395453900708937	0.7665
2019	MLB	3346	20.35393604303708	0.7218
2019	PGA	1305	22.249034482759008	1.1502
2019	NHL	1254	21.269545454545767	0.6842

MySQL Code:

```
WITH sports AS (
  SELECT
    YEAR(e.entry_week) AS entry_year,
    e.entry_sport,
    COUNT(e.total_entries) AS total_entries,
    AVG(u.first_deposit_amount) AS avg_first_deposit,
    AVG(DATEDIFF(u.first_paid_entry_date, u.registration_date)) AS
avg_days_to_first_entry,
    RANK() OVER (PARTITION BY YEAR(e.entry_week) ORDER BY
COUNT(e.total_entries) DESC) AS sport_rank
  FRoM
    entry e
  JOIN
    user u ON e.product_account_id = u.product_account_id
  WHERE
    YEAR(e.entry_week) IN (2018, 2019)
    AND DATEDIFF(u.first_paid_entry_date, u.registration_date) <= 7
    AND u.first_paid_entry_date IS NOT NULL
  GROUP BY
    YEAR(e.entry_week),
        e.entry_sport
)
SELECT
    entry_year,
    entry_sport,
    total_entries,
    avg_first_deposit,
    avg_days_to_first_entry
FROM
    sports
WHERE
    sport_rank <= 5
ORDER BY
    entry_year,
    sport_rank;
```