Wedding Videography Business Project

By: Austin Shirk

This project aims to provide valuable insights into revenue trends, employee activity, popular wedding video packages and other key metrics through a series of complex SQL queries.

Note: *Data was created by a randomizer. No data was taken from a real wedding videography business. *

The queries executed include the following:

1. Calculate total revenue by month for the latest year
2. Calculate total revenue for each package type over the past 12 months
3. Identify videographers who shot the most weddings for each of the past 3 years
4. Compare the number of weddings held in Q1 (January-March) and Q4 (October-December) for each year
5. Identify the month with the most weddings for each year
6. For each year, find the month with the highest and lowest number of bookings, along with the average package price for each month
7. Show email, phone number, full name, and wedding date for clients with Florida or Pennsylvania weddings
8. Identify the most popular wedding video package for each year
9. Identify the month with the highest MoM increase for each year

Tables:

```sql
CREATE TABLE customers (
customer_id INT,
first_name VARCHAR(255),
last_name VARCHAR(255),
home_state VARCHAR(255),
phone_number VARCHAR(255),
email VARCHAR(255),
PRIMARY KEY (customer_id)
);

CREATE TABLE team (
videographer_id INT,
first_name VARCHAR(255),
last_name VARCHAR(255),
PRIMARY KEY (videographer_id)
);

CREATE TABLE orders (
order_id INT,
customer_id INT,
package_type INT,
wedding_date VARCHAR(255),
booked_date VARCHAR(255),
order_amount FLOAT,
venue_state VARCHAR(255),
videographer_id INT,
videographer_amount FLOAT,
hours_of_coverage FLOAT,
PRIMARY KEY (order_id),
FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
FOREIGN KEY (videographer_id) REFERENCES team(videographer_id)
);
```

1. Calculate total revenue by month for the latest year, sorted by revenue in descending order

```sql
SELECT
    SUM(order_amount) AS 'Total Revenue',
    DATE_FORMAT(booked_date, '%M') AS month,
    YEAR(booked_date) AS year
FROM orders
WHERE YEAR(booked_date) = YEAR((SELECT MAX(booked_date) FROM orders))
GROUP BY 2, 3
ORDER BY SUM(order_amount) DESC
```

Output:

Total Revenue	month	year
51520	June	2023
45200	August	2023
41010	April	2023
38490	July	2023
35690	October	2023
35610	March	2023
35210	May	2023
34859	September	2023
32590	February	2023
28235	December	2023
26550	November	2023
22750	January	2023

2. **Calculate total revenue for each package type over the past 12 months**
    a. **sorted by revenue in descending order**

```sql
SELECT
        SUM(order_amount) AS order_amount,
        package_type
FROM orders
WHERE booked_date >= DATE_SUB((SELECT MAX(booked_date) FROM orders), INTERVAL 12 Month)
GROUP BY package_type
ORDER BY SUM(order_amount) DESC
```

Output:

package_type	Total Revenue
3	349105
2	82554

3. **Identify videographers who shot the most weddings for each of the past 3 years (based on wedding_date)**
    a. **Additionally, output the number of weddings and total videographer pay**

```sql
WITH CTE AS (
  SELECT
    t.first_name, t.last_name, o.videographer_id,
    COUNT(o.order_id) AS wedding_count,
    YEAR(o.wedding_date) AS year,
    SUM(o.videographer_amount) AS videographer_total_amount,
    RANK() OVER (PARTITION BY YEAR(o.wedding_date) ORDER BY COUNT(o.order_id) DESC) AS wedding_count_rank
  FROM orders o
  JOIN team t USING (videographer_id)
  WHERE YEAR(o.wedding_date) >= YEAR((SELECT MAX(wedding_date) FROM orders)) - 2
  GROUP BY videographer_id, YEAR(o.wedding_date)
)

SELECT year, first_name, last_name, videographer_id, wedding_count, videographer_total_amount
FROM CTE
WHERE wedding_count_rank = 1
ORDER BY year;
```

Output:

year	first_name	last_name	videographer_id	wedding_count	videographer_total_amou...
2022	Renee	E	9	25	30150
2023	Renee	E	9	15	17850
2024	Cameron	Z	6	26	34050

4. **Compare the number of weddings held in Q1 (January-March) and Q4 (October-December) for each year**

```sql
SELECT
    YEAR(wedding_date) AS 'Year',
    SUM(CASE WHEN MONTH(wedding_date) IN (1,2,3) THEN 1 ELSE 0 END) AS 'Q1 Wedding Count',
    SUM(CASE WHEN MONTH(wedding_date) IN (10,11,12) THEN 1 ELSE 0 END) AS 'Q4 Wedding Count'
FROM
    orders
GROUP BY YEAR(wedding_date)
ORDER BY 1;
```

Output:

Year	Q1 Wedding Count	Q4 Wedding Count
2021	17	22
2022	17	19
2023	14	16
2024	13	28

5. **Identify the month with the most weddings for each year**

```sql
WITH CTE AS (
    SELECT
        COUNT(order_id) AS wedding_count,
        RANK() OVER (PARTITION BY YEAR(wedding_date) ORDER BY COUNT(order_id) DESC) AS count_ranking,
        MONTH(wedding_date) AS month,
        YEAR(wedding_date) AS year
    FROM
    orders
    GROUP BY MONTH(wedding_date), YEAR(wedding_date)
)

SELECT wedding_count AS "Wedding Count", month AS "Month", year AS "Year"
FROM CTE
WHERE count_ranking = 1;
```

Output:

Wedding Count	Month	Year
13	7	2021
12	7	2022
11	7	2023
14	10	2024

6. **For each year, find the month with the highest and lowest number of bookings, along with the average package price for each month.**

```sql
WITH CTE AS (
    SELECT
        DATE_FORMAT(booked_date, '%M') AS month,
        YEAR(booked_date) AS year,
        RANK() OVER (PARTITION BY YEAR(booked_date) ORDER BY COUNT(order_id) DESC) as count_ranking,
        AVG(order_amount) AS avg_order_amount,
        COUNT(order_id) AS booking_count
    FROM orders
    GROUP BY month, YEAR(booked_date)
)

SELECT
    year,
    MAX(booking_count) OVER (PARTITION BY year) AS max_month_bookings,
    MIN(booking_count) OVER (PARTITION BY year) AS min_month_bookings,
    CASE WHEN count_ranking = 1 THEN month END AS highest_booking_month,
    CASE WHEN count_ranking = 12 THEN month END AS lowest_booking_month,
    ROUND(avg_order_amount, 2) AS avg_package_price
FROM CTE
WHERE count_ranking IN (1, 12);
```

Output:

year	max_month_bookings	min_month_bookings	highest_booking_month	lowest_booking_month	avg_package_price
2020	11	6	June	NULL	1991.82
2020	11	6	NULL	February	2060.83
2021	11	4	June	NULL	3605.91
2021	11	4	August	NULL	2800
2021	11	4	NULL	February	3401.25
2022	11	11	April	NULL	3994.09
2023	11	5	June	NULL	4683.64
2023	11	5	NULL	January	4550

7. **Show email, phone number, full name, and wedding date for clients with Florida or Pennsylvania weddings, sorted by state then first_name (ascending).**

```
123 •   SELECT c.first_name, c.last_name, c.email, c.phone_number, o.wedding_date, o.venue_state
124     FROM orders o
125     JOIN customers c USING (customer_id)
126     WHERE o.venue_state IN ('Florida', 'Pennsylvania')
127     ORDER BY o.venue_state, c.first_name;
```

Output:

first_name	last_name	email	phone_number	wedding_date	venue_state
Alex	B	test159@example.com	555-1158	2022-08-12	Florida
Alexander	W	test25@example.com	555-1024	2021-04-03	Florida
Alexander	G	test30@example.com	555-1029	2021-04-28	Florida
Ava	M	test184@example.com	555-1183	2022-06-09	Florida
Ava	C	test61@example.com	555-1060	2021-07-01	Florida
Avery	J	test5@example.com	555-1004	2021-01-12	Florida
Benjamin	T	test27@example.com	555-1026	2021-04-24	Florida
Benjamin	H	test84@example.com	555-1083	2021-10-28	Florida
David	B	test100@example.com	555-1099	2021-12-31	Florida
David	H	test59@example.com	555-1058	2021-07-08	Florida
Eleanor	B	test113@example.com	555-1112	2022-03-31	Florida
Eleanor	J	test98@example.com	555-1097	2021-09-26	Florida
Ethan	S	test247@example.com	555-1246	2023-01-21	Florida
Evelyn	L	test174@example.com	555-1173	2022-10-07	Florida
Finn	H	test234@example.com	555-1233	2023-12-16	Florida
Finn	J	test251@example.com	555-1250	2023-07-30	Florida
Isla	B	test104@example.com	555-1103	2022-06-04	Florida
Isla	W	test43@example.com	555-1042	2021-06-30	Florida
Isla	L	test237@example.com	555-1236	2023-11-04	Florida
Luna	A	test129@example.com	555-1128	2022-05-14	Florida
Penelope	M	test115@example.com	555-1114	2022-03-18	Florida
Penelope	J	test339@example.com	555-1338	2024-09-06	Florida
Rachel	N	test361@example.com	555-1360	2024-12-06	Florida
Sadie	G	test69@example.com	555-1068	2021-09-04	Florida
Sadie	L	test16@example.com	555-1015	2021-03-19	Florida
Scarlett	H	test252@example.com	555-1251	2023-02-24	Florida
Sebastian	W	test205@example.com	555-1204	2023-06-03	Florida
Sofia	A	test93@example.com	555-1092	2021-12-10	Florida
William	M	test73@example.com	555-1072	2021-09-18	Florida
Christopher	D	test94@example.com	555-1093	2021-07-08	Pennsylvania
Leo	A	test240@example.com	555-1239	2023-09-07	Pennsylvania
Riley	W	test145@example.com	555-1144	2022-07-01	Pennsylvania
Scarlett	M	test186@example.com	555-1185	2022-07-09	Pennsylvania
Violet	A	test47@example.com	555-1046	2021-06-11	Pennsylvania

8. Identify the most popular wedding video package for each year

a. Display the package name instead of the package_type

Package Key:

package_type	package_name
1	The Memoir
2	The Standard
3	The Luxury

```sql
WITH CTE AS (
    SELECT
        COUNT(order_id) AS wedding_count,
        CASE
            WHEN package_type = 1 THEN 'The Memoir'
            WHEN package_type = 2 THEN 'The Standard'
            WHEN package_type = 3 THEN 'The Luxury'
            END AS package_name,
        RANK() OVER (PARTITION BY YEAR(booked_date) ORDER BY COUNT(order_id) DESC) AS package_rank,
        YEAR(booked_date) AS year
    FROM orders
    GROUP BY package_type, YEAR(booked_date)
    ORDER BY YEAR(booked_date) DESC
)
SELECT package_name AS 'Package Name', wedding_count AS 'Number of Weddings', year AS 'Year'
FROM CTE
WHERE package_rank = 1;
```

Output:

	Package Name	Number of Weddings	Year
	The Luxury	74	2023
	The Standard	75	2022
	The Standard	46	2021
	The Memoir	79	2020

9. Identify the month with the highest MoM increase for each year

MoM formula:

MoM = month_total_package_amount - previous_month_total_package_amount

```sql
WITH MonthlyTotalAmount AS (
    SELECT
        SUM(order_amount) total_month_amount,
        month(booked_date) AS month,
        year(booked_date) AS year
    FROM orders
    GROUP BY year(booked_date), month(booked_date)
    ORDER BY year(booked_date), month(booked_date)
),
MoMComparison AS (
    SELECT
        total_month_amount AS month_total,
        lag(total_month_amount) OVER () AS previous_month_total,
        year, month,
        total_month_amount-lag(total_month_amount) OVER () AS mom
    FROM MonthlyTotalAmount
)

SELECT month, year, mom AS 'MoM Increase', month_total, previous_month_total
FROM
    (SELECT
        RANK() OVER (PARTITION BY year ORDER BY mom DESC) as mom_rank,
        mom, year, month, month_total, previous_month_total
    FROM MoMComparison) as subquery
WHERE mom_rank = 1
```

Output:

	month	year	MoM Increase	month_total	previous_month_total
	6	2020	5350	21910	16560
	3	2021	12915	26520	13605
	4	2022	20315	43935	23620
	6	2023	16310	51520	35210