Brianna Griffin
MAT 265

New MySQL Commands

Introduction: For this week's assignment, I utilized the couse data base and SPL data to experiment and teach myself new MySQL functions and commands. I was able to learn new aggregate, time, and string functions that furthered my knowledge of the SQL programming language while discovering new findings within the database.

1. Query # 1: For the first query, I looked into the many aggregate functions that SQL has. I used the following functions within my query in order to calculate statistics on the number of CD checkouts during 2022:
    a. STDDEV() : calculates the standard deviation
    b. COUNT() : counts the number of rows
    c. AVG() : calculates the average
    d. VARIANCE() : calculates the variance
    e. FORMAT() : to determine how I wanted to round the calculated statistics

---

```
SELECT
   FORMAT(AVG(num_couts), 0) AS average,
   FORMAT(STDDEV(num_couts), 0) AS SD,
   FORMAT(VARIANCE(num_couts), 0) AS variance_cd_COs
FROM
   (SELECT
      COUNT(cout) AS num_couts, MONTH(cout) AS month_of_couts
   FROM
      spl_2016.outraw
   WHERE
      itemtype LIKE '%cd'
         AND YEAR(cout) = 2020
   GROUP BY 2
   ORDER BY 2 ASC) sub;
```

The output table (below) displays the average number of monthly CD checkouts in 2022 along with the standard deviation and variation. It makes sense that the standard deviation is so high because the pandemic hit during 2020. Therefore, checkouts went from very high to nothing because everyone was locked down due to Covid-19. The variance is the standard deviation squared.

average	SD	variance_cd_COs
10,792	10,413	108,437,072

2. Query # 2: In this query, I experimented with date and time functions in SQL. Specifically, I used new functions to see what times (minutes and hour) people checked out an item from the SPL two months ago from today. I utilized the below SQL functions:
    a. DATE_FORMAT(): reference
        i. %H:%i = hour : minute (24 hours i.e. military time)
        ii. Used to format the output for clarity and analysis
    b. DATE(cout): takes month, date, and year from cout time in data base
    c. CURRENT_DATE(): outputs today's date
    d. DATE_SUB(x, y): subtracts a time interval(y) from a date(x)

```
SELECT DISTINCT
   (DATE_FORMAT(cout, '%H:%i')) AS times
FROM
   spl_2016.outraw
WHERE
   DATE(cout) = DATE_SUB(CURRENT_DATE(),
      INTERVAL 2 MONTH);
```

- OUTPUT: here
    - The output shows that on September 7, 2022 (2 months from the day I ran the query), people were checking out items at a steady rate. There was rarely a gap larger than 5 minutes throughout the day where an item was not checked out up until 7:59. This is likely when the library closed for the day.
    - The date format function was really easy to use and is useful to me since this quarter I have worked with checkout and check in times so much and the raw data is not in a very usable format.

3. Query # 3: Next, I am looking at string functions via the titles of items at SPL. First, I wanted to see what the SOUNDS LIKE function does in a SQL query. I inputted this into a query to see if there are any titles that "sound like 'car' ". Per the description the sounds like function will compare sounds so I assumed that the output would contain rhyming sounds, but I more so found that the outputted titles shared the first letter of 'c' with car.

a. I also used the function CHAR_LENGTH() to return titles of at least length 3 for relevence purposes.

```
SELECT DISTINCT
   (title)
FROM
   spl_2016.inraw
WHERE
   title SOUNDS LIKE SOUNDEX('car')
      AND CHAR_LENGTH(title) >= 3
      AND title NOT LIKE '%car%';
```

- OUTPUT: here
- ★ When querying for results that do not start with ‘c’, there are only 3 outputs. Those 3 are titles with ‘c’ as the beginning excluded numbers before then.
  - Here is the query and output table for this:

```
SELECT DISTINCT
      (title)
FROM
   spl_2016.inraw
WHERE
   title SOUNDS LIKE SOUNDEX('car')
      AND CHAR_LENGTH(title) >= 3
      AND title NOT LIKE '%car%'
      AND LEFT(title, 1) != 'c';
```

---

4. Query # 4: For the last query, I also used string functions in order to see how many items start with each letter of the alphabet. I used the substring function to pull the first letter from the title of each item. I also disregarded items that start with anything other than a letter. There is a very long case-when statement that basically just divides the items into categories based on the letter that they start with.

```
SELECT
   COUNT(title) AS frequency,
   CASE
```

```
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'a' THEN 'A'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'b' THEN 'B'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'c' THEN 'C'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'd' THEN 'D'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'e' THEN 'E'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'f' THEN 'F'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'g' THEN 'G'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'h' THEN 'H'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'i' THEN 'I'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'j' THEN 'J'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'k' THEN 'K'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'l' THEN 'L'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'm' THEN 'M'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'n' THEN 'N'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'o' THEN 'O'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'p' THEN 'P'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'q' THEN 'Q'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'r' THEN 'R'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 's' THEN 'S'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 't' THEN 'T'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'u' THEN 'U'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'v' THEN 'V'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'w' THEN 'W'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'x' THEN 'X'
      WHEN LOWER(SUBSTRING(title, 1, 1)) = 'y' THEN 'Y'
      ELSE 'Z'
   END AS starts_with
FROM
   (SELECT DISTINCT
      (title)
   FROM
      spl_2016.outraw
   WHERE
      CHARACTER_LENGTH(title) >= 1
         AND SUBSTRING(title, 1, 1) NOT IN ('^' , '-', '?', '+', '%', '&', '$', '0', '1', '2', '3', '4', '5', '6',
'7', '8', '9')) sub
GROUP BY 2
```

- OUTPUT: The output table (below) shows that items beginning with I are the most frequent and beginning with X are the least frequent.

frequency	starts_with
47448	A
53160	B
60235	C
38797	D
26189	E
35546	F
31283	G
39075	H
114898	I
13434	J
13318	K
36852	L
55524	M
23916	N
19309	O
42665	P
3724	Q
30948	R
78518	S
43251	T
9899	U
10650	V
39644	W
1580	X
7956	Y
3409	Z

Conclusion: I learned a lot from this assignment because it allowed me to work outside of my comfort zone, experimenting with new SQL functions. I will use these in the future. It also allowed me to explore new topics within the SPL course database.