SQL Workshop (Dec): Introduction to SQL – Day 3

RECAP

1. Given the following syntaxs:
   a. SELECT, LIMIT, ORDER BY, FROM, WHERE
   b. How should they be placed?

2. How do I select the customer information for those whose first_name contains ‘a’ and have ‘e’ as the fourth character from the back?

3. If I only want 10 rows, what do I do?

4. Write the line of code to arrange col1 by ascending, col2 by asc, col3 by desc

Aggregate Functions

COUNT	Return the number of rows in the stated column
SUM	Return the total value of the stated numeric column
AVG	Return the average value of the stated numeric column
MIN	Return the lowest value in the stated column
MAX	Return the highest value in the stated column

COUNT

```
SELECT COUNT(*) FROM sales.customers
```

	count bigint
1	1445

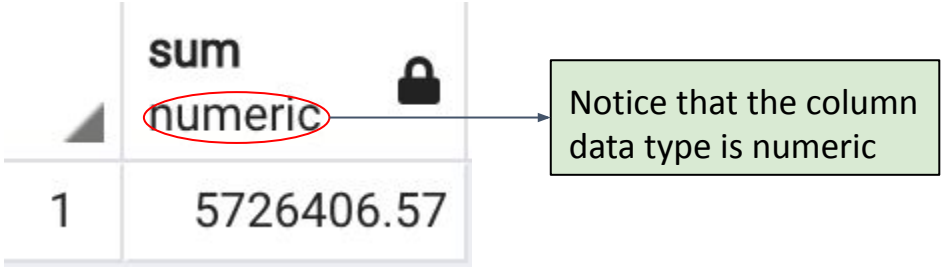
"DISTINCT" can also be used with aggregate functions

```
SELECT COUNT(DISTINCT city) FROM sales.customers
```

	count bigint
1	195

SUM

```
SELECT SUM(list_price) FROM sales.order_items
```




```
SELECT SUM(list_price) FROM sales.order_items
WHERE list_price > 599.99
```

	sum numeric
1	4646445.21

AVG

```
SELECT AVG(discount) FROM sales.order_items
```

	avg numeric
1	0.10537272342227869547

```
SELECT AVG(discount) AS average_discount FROM sales.order_items
```

	average_discount numeric
1	0.10537272342227869547

"AS" allows an alias name to be given for the duration of the query, can be used for table and column

Alias name can also be represented without "AS" e.g. AVG(discount) avg_discount or sales.order_items OI

MIN

```
SELECT MIN(discount) FROM sales.order_items
```

	min numeric
1	0.05

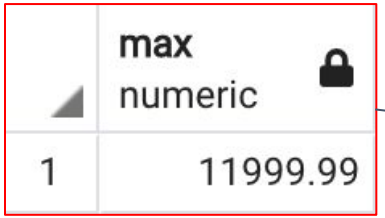
If no conditions were stated, return value by default will be from all the rows of the stated column

```
SELECT MIN(discount) FROM sales.order_items
WHERE list_price = 699.99
```

	min numeric
1	0.10

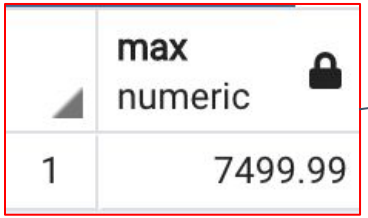
MAX

```
SELECT MAX(list_price) FROM sales.order_items
```



	max numeric
1	11999.99

```
SELECT MAX(list_price) FROM sales.order_items
WHERE discount < 0.07
```



	max numeric
1	7499.99

Remember, stating conditions can change the returned value

HANDS-ON (Aggregate Functions)

1. What is the number of unique list_price and unique discount?

2. What is the maximum discount and minimum list_price?

3. Using an appropriate SQL statement, find the average number of quantity, avg_qty, in store_id 3 for product_id 10,15 and 20

4. Using an appropriate SQL statement, find the unique number of customers who ordered from store_id 2, in the year 2017 or on the required_date “2016-02-19” OR “2016-03-11”

GROUP BY

- Often used with aggregate functions
- Group rows with the same values together
  - Like you have a column of STATES – which means there can be Texas (TX), California (CA),
  - You just want to find the greatest value of a single state, instead of among all the states combined.

- Think of this as Pivot Table from Excel!

GROUP BY

state character varying (25)
NY
CA
CA
NY
CA
NY
NY
NY
NY
NY
NY
NY
NY
TX

Do you noticed something about this column?

GROUP BY

```
SELECT COUNT(customer_id) AS num_cust, state FROM
sales.customers GROUP BY state
```

	num_cust bigint	state character varying (25)
1	1019	NY
2	142	TX
3	284	CA

Can be represented by the column name “state” or the column number “2”

```
SELECT COUNT(customer_id) AS num_cust, state FROM
sales.customers
```

```
ERROR:  column "customers.state" must appear in the GROUP BY clause or be used in an aggregate function
LINE 1: SELECT count(customer_id) AS num_cust, state FROM sales.cust...
                                               ^
```

HAVING - Similar to WHERE

Same function as WHERE but used with aggregate functions

```
SELECT SUM(list_price)
FROM sales.order_items
WHERE list_price > 599.99
```

```
SELECT COUNT(customer_id) AS
num_cust, state
FROM sales.customers
GROUP BY state
HAVING COUNT(customer_id) > 200
```

You can still use WHERE when you used GROUP BY + HAVING!

Derived Column

- Obtains a new column through the use of math operators
  - Order of operations follows the **PEMDAS** rules

Derived Column

SELECT list_price, discount, (list_price * discount) AS discounted_price FROM sales.order_items

	list_price numeric (10,2)	discount numeric (4,2)	discounted_price numeric
1	599.99	0.20	119.9980
2	1799.99	0.07	125.9993
3	1549.00	0.05	77.4500
4	599.99	0.05	29.9995
5	2899.99	0.20	579.9980

SELECT discount * 100 AS percentage_off FROM sales.order_items

	percentage_off numeric
1	20.00
2	7.00
3	5.00
4	5.00
5	20.00

HANDS-ON (Aggregate Functions - 2)

1. Using an appropriate SQL statement, find the number of unique customers, num_cust, from each store that have received their order. (shipped_date)[orders]

2. Using an appropriate SQL statement, find the staff_id that is responsible for more than 50 late deliveries from store_id 1 (dates)[orders]

3. Using an appropriate SQL statement, find the store_id of the store(s) that have at least more than 10 product_id with quantity 0 (total number of product_id is 313)[stocks]

4. Using an appropriate SQL statement, find the top 5 most expensive average price after discount, avg_price_after_discount, of each quantity by their product_id for product_id 1 to 15 [order_items]

END OF DAY 3!

Any questions? Feel free to clarify now.

Or you can reach us at:

Wei Teck (Jensen): wtlow003@suss.edu.sg / https://www.linkedin.com/in/weitecklow/

Jeanette: jeanettekoh001@suss.edu.sg / linkedin.com/in/jeanette-koh-872b64192