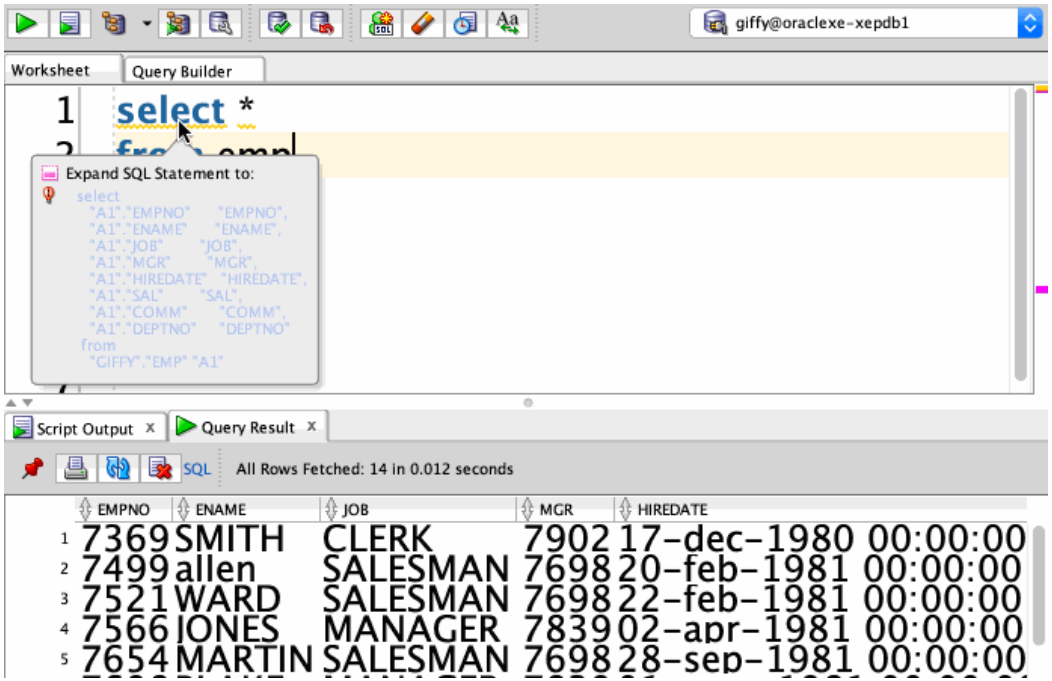
CHAPTER 3B : More on SQL

XIAMEN UNIVERSITY MALAYSIA
厦門大學馬來西亞分校

Outline

- More on SQL Syntax



WHERE (1)

Find all the information about Customers who is above 20 years old.

```
SELECT *
FROM CUSTOMERS
WHERE AGE > 20;
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 6 in 0.003 seconds

	CID	CNAME	AGE	EMAIL
1	3	Jacenta	28	(null)
2	4	Joanne	40	(null)
3	5	Clement	33	(null)
4	6	Yap	25	(null)
5	7	Clovie	80	(null)
6	8	James	60	(null)

WHERE (2)

Find the title, author and year where the published year is 2000 and below.

```
SELECT TITLE, AUTHOR, YEAR
FROM BOOKS
WHERE YEAR <= 2000;
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 4 in 0.002 seconds

	TITLE	AUTHOR	YEAR
1	Introduction to Databases	David Maier	1981
2	Blindness	Jose Saramago	1999
3	Principles of Database and Knowledge-Base Systems	Jeffrey Ullman	1997
4	The Gospel According to Jesus Christ	Jose Saramago	1994

DISTINCT

Use to eliminate the duplicates of data.

```
SELECT DISTINCT CID
FROM Customers;
```

Script Output | Query Result

SQL | All Rows Fetched: 8 in 0.003 seconds

	CID
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8

AND, OR and NOT

AND

- If all of the **conditions separated by AND are TRUE**, the AND operator displays a record.

OR

- If any of the **conditions separated by OR is TRUE**, the OR operator displays a record.

NOT

- If the **condition(s) are NOT TRUE**, the NOT operator shows a record.

AND (1)

Find the information about customer who is above 20 years and has identification number 3 and below.

```
SELECT *
FROM CUSTOMERS
WHERE AGE > 20
AND CID <= 3;
```

Script Output | Query Result

SQL | All Rows Fetched: 6 in 0.003 seconds

	CID	CNAME	AGE	EMAIL
1	3	Jacenta	28	(null)
2	4	Joanne	40	(null)
3	5	Clement	33	(null)
4	6	Yap	25	(null)
5	7	Clovie	80	(null)
6	8	James	60	(null)

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.003 seconds

	CID	CNAME	AGE	EMAIL
1	3	Jacenta	28	(null)

AND (2)

With Join query.

Find the information about the book's author, title and purchase date of the book which is written in year 1981.

```
SELECT TITLE, AUTHOR, PDATE
FROM PURCHASES, BOOKS
WHERE PURCHASES.BOOKID = BOOKS.BOOKID
AND YEAR = '1981';
```

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.003 seconds

	TITLE	AUTHOR	PDATE
1	Introduction to Databases	David Maier	10-JAN-19
2	Introduction to Databases	David Maier	10-FEB-12

OR (1)

Find all the information about either the customer's identification number 3 or else their age is 40.

```
SELECT *
FROM CUSTOMERS
WHERE CID = 3
OR AGE = 40;
```

Script Output | Query Result

SQL | All Rows Fetched: 6 in 0.003 seconds

	CID	CNAME	AGE	EMAIL
1	3	Jacenta	28	(null)
2	4	Joanne	40	(null)
3	5	Clement	33	(null)
4	6	Yap	25	(null)
5	7	Clovie	80	(null)
6	8	James	60	(null)

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.003 seconds

	CID	CNAME	AGE	EMAIL
1	3	Jacenta	28	(null)
2	4	Joanne	40	(null)

NOT

Find all the information about the customers whose is the age is other than 40 years old.

```
SELECT *
FROM CUSTOMERS
WHERE NOT AGE = 40;
```

Script Output | Query Result

SQL | All Rows Fetched: 6 in 0.003 seconds

	CID	CNAME	AGE	EMAIL
1	3	Jacenta	28	(null)
2	4	Joanne	40	(null)
3	5	Clement	33	(null)
4	6	Yap	25	(null)
5	7	Clovie	80	(null)
6	8	James	60	(null)

Script Output | Query Result

SQL | All Rows Fetched: 7 in 0.002 seconds

CID	CNAME	AGE	EMAIL
3	Jacenta	28	(null)
5	Clement	33	(null)
6	Yap	25	(null)
7	Clovie	80	(null)
8	James	60	(null)

SQL ORDER BY Keyword

- To sort the result set in ascending or descending order, use the **ORDER BY** keyword.
- By default, the ORDER BY keyword sorts the records in ascending order. Use the **DESC** keyword to sort the records in descending order.

```
SELECT *
FROM PRICING
ORDER BY PRICE;
```

Script Output | Query Result

SQL | All Rows Fetched: 9 in 0.004 seconds

	PRICEREF	BOOKID	FORMAT	PRICE
1	908765	106	paperback	7.48
2	564321	105	paperback	12.55
3	564390	108	paperback	14.33
4	789000	107	paperback	17.98
5	123456	101	hardcover	32.95
6	345678	103	paperback	35
7	555201	109	paperback	44.88
8	134567	102	hardcover	120
9	890765	104	hardcover	130

```
SELECT *
FROM PRICING
ORDER BY PRICE DESC;
```

Script Output | Query Result

SQL | All Rows Fetched: 9 in 0.002 seconds

	PRICEREF	BOOKID	FORMAT	PRICE
1	890765	104	hardcover	130
2	134567	102	hardcover	120
3	555201	109	paperback	44.88
4	345678	103	paperback	35
5	123456	101	hardcover	32.95
6	789000	107	paperback	17.98
7	564390	108	paperback	14.33
8	564321	105	paperback	12.55
9	908765	106	paperback	7.48

ORDERBY (JOIN QUERY)

To find all the information about the pricing and purchases which the price is lower than 10, and the outcomes of the price should be in the ascending order.

```
SELECT *
FROM PRICING, PURCHASES
WHERE PRICING.BOOKID = PURCHASES.BOO
AND PRICE < 10
ORDER BY PRICING.PRICE;
```

Script Output | Query Result

SQL | All Rows Fetched: 9 in 0.004 seconds

	PRICEREF	BOOKID	FORMAT	PRICE
1	908765	106	paperback	7.48
2	564321	105	paperback	12.55
3	564390	108	paperback	14.33
4	789000	107	paperback	17.98
5	123456	101	hardcover	32.95
6	345678	103	paperback	35
7	555201	109	paperback	44.88
8	134567	102	hardcover	120
9	890765	104	hardcover	130

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.002 seconds

	PRICEREF	BOOKID	FORMAT	PRICE	CID	BOOKID_1	PDATE
1	908765	106	paperback	7.48	1	106	01-JUL-18

UPDATE

To update the existing customer's information whose the ID is 3.

UPDATE **CUSTOMERS**

SET **CName = 'SUBASHINI'**

WHERE **CID = 3;**

Then type in this command,

SELECT * FROM CUSTOMERS;

Script Output | Query Result

SQL | All Rows Fetched: 8 in 0.001 seconds

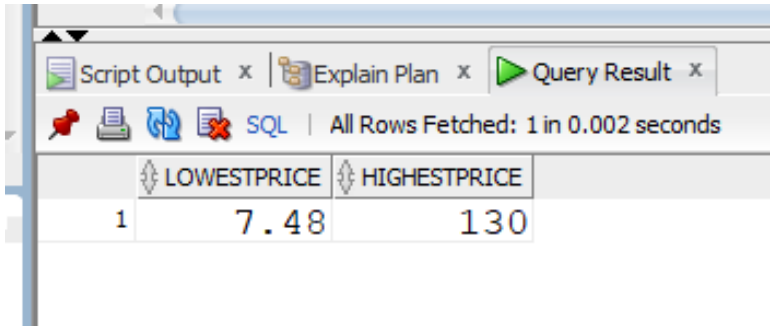
	CID	CNAME	AGE	EMAIL
1	1	Diana	20	(null)
2	2	Thomas	18	(null)
3	3	SUBASHINI	28	(null)
4	4	Joanne	40	(null)
5	5	Clement	33	(null)
6	6	Yap	25	(null)
7	7	Clovie	80	(null)
8	8	James	60	(null)

MIN and MAX

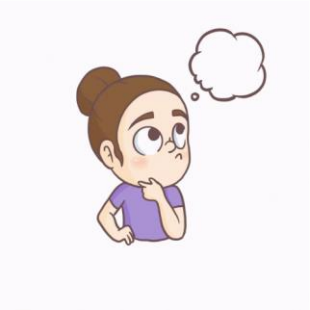
The MIN() function returns the column's smallest value.

The MAX() function returns the column's maximum value.

```
SELECT MIN(PRICE) AS LOWESTPRICE,
       MAX(PRICE) AS HIGHESTPRICE
FROM   PRICING;
```



AVG and SUM



Can you find query example for showing the AVG and SUM?

ROWNUM

In Oracle, the ROWNUM function is referred to as a pseudo-column.

Example, to retrieve a total of 5 row number of a specific table as shown below.

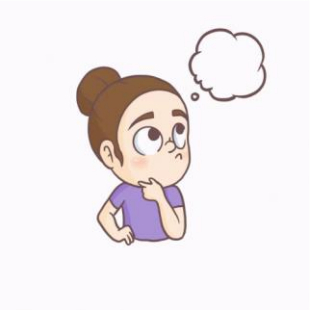
```
SELECT * FROM Customers
WHERE ROWNUM <= 5;
```

ry Result

SQL | All Rows Fetched: 5 in 0.002 seconds

CID	CNAME	AGE	EMAIL
1	Diana	20	(null)
2	Thomas	18	(null)
3	SUBASHINI	28	(null)
4	Joanne	40	(null)
5	Clement	33	(null)

LIKE



Can you find query example for showing the **LIKE?**

ADD COLUMN / DROP COLUMN

Lets say we are going to **alter the existing table CUSTOMERS** with **adding column** "Address", we use the following commands,

```
10 ALTER TABLE CUSTOMERS
11 ADD ADDRESS VARCHAR(25);
12
13
```

Script Output | Task completed in 0.03 seconds

```
Table CUSTOMERS altered.
```

SELECT *
FROM **CUSTOMERS**;

Script Output | Query Result | SQL | All Rows Fetched: 8 in 0.003 seconds

	CID	CNAME	AGE	ADDRESS
1	1	Diana	20	(null)
2	2	Thomas	18	(null)
3	3	SUBASHINI	28	(null)
4	4	Joanne	40	(null)
5	5	Clement	33	(null)
6	6	Yap	25	(null)
7	7	Clovie	80	(null)
8	8	James	60	(null)

ADD COLUMN / DROP COLUMN

Lets say we are going to **alter the existing table CUSTOMERS** with **dropping column** "Address", we use the following commands,

```
20 ALTER TABLE CUSTOMERS
21 DROP COLUMN ADDRESS;
22
```

Script Output | Query Result

Task completed in 0.024 seconds

```
Table CUSTOMERS altered.
```

SELECT *
FROM **CUSTOMERS**;

Script Output | Query Result

SQL | All Rows Fetched: 8 in 0.003 seconds

	CID	CNAME	AGE
1	1	Diana	20
2	2	Thomas	18
3	3	SUBASHINI	28
4	4	Joanne	40
5	5	Clement	33
6	6	Yap	25
7	7	Clovie	80
8	8	James	60

UNION

- To combine the results of two or more SELECT statements, use the UNION operator.
  - Within UNION, every SELECT statement must have the same number of columns.
  - The data types in the columns must also be similar.

Example, to select year (>2000) column from Books table and combine them with price column from pricing table which has price lower than 15.

```
23 SELECT YEAR
24 FROM BOOKS
25 WHERE YEAR > 2000
26 UNION
27 SELECT PRICE
28 FROM PRICING
29 WHERE PRICE < 15;
30
31
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.002 seconds

	YEAR
1	7.48
2	12.55
3	14.33
4	2004
5	2008

SQL CASE

- When the **first condition is met**, the CASE statement goes through the conditions and returns a value (like an **if-then-else statement**).
- When a **condition is true**, the **program will stop reading and return the result**.
  - It returns the value in the **ELSE clause if none of the conditions are true**.
  - It returns **NULL if there is no ELSE part and no conditions are true**.

Example, lets say we want to retrieve and categories the price value which is above 100, exactly 100 and below than 100.

We can use CASE statement to represent the value.

```sql
CASE
    WHEN PRICE > 100 THEN 'The price is greater than 100'
    WHEN PRICE = 100 THEN 'The price is 100'
    ELSE 'The price is under 100'
END AS PRICE_DESCRIPTION
FROM PRICING;
```

t Output × | Query Result ×

SQL | All Rows Fetched: 9 in 0.003 seconds

BOOKID	PRICE	PRICE_DESCRIPTION
101	32.95	The price is under 100
103	35	The price is under 100
104	130	The price is greater than 100
105	12.55	The price is under 100
107	17.98	The price is under 100
108	14.33	The price is under 100
109	44.88	The price is under 100
102	120	The price is greater than 100
106	7.48	The price is under 100

Practice More

You may refer and practice more SQL syntax by referring to this link,

https://www.w3schools.com/sql/default.asp

https://www.techonthenet.com/oracle/index.php

https://www.khanacademy.org/computing/computer-programming/sql

https://www.shiksha.com/online-courses/articles/how-to-use-where-clause-in-sql/