SQL Review

Lecture 01

Agenda

- SELECT
- JOIN

SELECT

Fetch Data from a Database

SQL Statements

Data Manipulation Language (DML)	SELECT INSERT UPDATE DELETE
Data Definition Language (DDL)	CREATE ALTER DROP RENAME
Data Control Language (DCL)	GRANT REVOKE
Transaction Control Language (TCL)	COMMIT ROLLBACK SAVEPOINT

Select

```
SELECT [distinct]
  column1,
  column2,
  [expression]
  ...
FROM
  tablename
[WHERE conditions];
```

- SELECT: identifies the columns or calculations
- FROM: identifies the table that you want to retrieve data from
- WHERE: identifies the condition that has to be met for the record to be selected.

Select

```
SELECT customer_id, name, address
FROM   customers;
```

- The above query shows actor_id, first_name, and last_name for all actors in table **customers**.

```
SELECT customer_id, name, address
FROM   customers
WHERE first_name = 'NICK';
```

- The above query shows actor_id, first_name, and last_name for actors in table **customer** whose first_name is *NICK*.

```
SELECT *                    -- selecting all
columns
FROM   customers;
```

Column Alias

```
SELECT list_price * 1.05 as new_price
FROM products;
```

```
SELECT list_price * 1.05 as "New Price"
FROM products;
```

	NEW_PRICE
1	3580.983
2	2913.729
3	2793.756
4	2682.7395
5	2626.7745
6	2553.5475
7	2495.9445
8	2383.4895

...

	New Price
1	3580.983
2	2913.729
3	2793.756
4	2682.7395
5	2626.7745
6	2553.5475
7	2495.9445
8	2383.4895

...

SORTING

- ORDER BY column_1[,column_2,...]
  - Is used to sort the result of a SQL select statement.
  - The default order is ascending.

```
SELECT *
FROM ass_product_categories
ORDER BY category_name;
```

	CATEGORY_ID	CATEGORY_NAME
1	1	CPU
2	4	Mother Board
3	3	RAM
4	5	Storage
5	2	Video Card

- Sorting in Descending Order
  - Use keyword DESC in the ORDER BY clause

```
SELECT *
FROM ass_product_categories
ORDER BY category_name DESC;
```

	CATEGORY_ID	CATEGORY_NAME
1	2	Video Card
2	5	Storage
3	3	RAM
4	4	Mother Board
5	1	CPU

Sorting By Column Alias

- The result of a SQL statement can be sorted by a column alias.

```
SELECT list_price * 1.05 as new_price
FROM ass_products
ORDER BY new_price;
```

	NEW_PRICE
1	16.3275
2	17.8395
3	28.3395
4	44.0895
5	46.1895
6	50.274
7	51.8385
8	55.2825
9	57.7395
10	60.879

```
SELECT list_price * 1.05 as "New Price"
FROM ass_products
ORDER BY "New Price";
```

	New Price
1	16.3275
2	17.8395
3	28.3395
4	44.0895
5	46.1895
6	50.274
7	51.8385
8	55.2825
9	57.7395
10	60.879

Sorting By Column Numeric Position

```
SELECT *
FROM ass_product_categories
ORDER BY 2;
```

	CATEGORY_ID	CATEGORY_NAME
1	1	CPU
2	4	Mother Board
3	3	RAM
4	5	Storage
5	2	Video Card

- The result of the above query is sorted based on the value of the second column.

```
SELECT *
FROM ass_product_categories
ORDER BY 1;
```

	CATEGORY_ID	CATEGORY_NAME
1	1	CPU
2	2	Video Card
3	3	RAM
4	4	Mother Board
5	5	Storage

- The result of the above query is sorted based on the value of the first column.

Sorting By Multiple Columns

```
SELECT *
FROM ass_warehouses
ORDER BY warehouse_name, location_id DESC;
```

- The result of the above query is sorted first by column warehouse_name ascendingly and then by column location_id descendingly.

	WAREHOUSE_ID	WAREHOUSE_NAME	LOCATION_ID
1	8	Beijing	11
2	9	Bombay	12
3	7	Mexico City	23
4	3	New Jersey	7
5	10	New Wharehouse	(null)
6	2	San Francisco	6
7	4	Seattle, Washington	8
8	1	Southlake, Texas	5
9	6	Sydney	13
10	5	Toronto	9

Simple Arithmetic Expressions

Operator	Description	Example
+	Add	SELECT list_price + 10 FROM products;
-	Subtract	SELECT list_price - 5 FROM products;
*	Multiply	SELECT list_price * 1.05 FROM products;
/	Divide	SELECT list_price / 2 FROM products;

	LIST_PRICE	LIST_PRICE+10	LIST_PRICE-5	LIST_PRICE*1.05	LIST_PRICE/2	LIST_PRICE*2+5	LIST_PRICE*(2+1)
1	3410.46	3420.46	3405.46	3580.983	1705.23	6825.92	10231.38
2	2774.98	2784.98	2769.98	2913.729	1387.49	5554.96	8324.94
3	2660.72	2670.72	2655.72	2793.756	1330.36	5326.44	7982.16
4	2554.99	2564.99	2549.99	2682.7395	1277.495	5114.98	7664.97
5	2501.69	2511.69	2496.69	2626.7745	1250.845	5008.38	7505.07
6	2431.95	2441.95	2426.95	2553.5475	1215.975	4868.9	7295.85
7	2377.09	2387.09	2372.09	2495.9445	1188.545	4759.18	7131.27
8	2269.99	2279.99	2264.99	2383.4895	1134.995	4544.98	6809.97

...

Operator Precedence

* /	Multiplication, division
+ -	Addition, subtraction

Comparison Operators and Conditions

Operator	Description
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<> (!=)	Not equal to
LIKE / NOT LIKE	Match a character pattern
BETWEEN / NOT BETWEEN	Checks values within a given range (inclusive)
IN / NOT IN	Checks multiple distinct values
IS NULL / IS NOT NULL	Checks if a column is null

LIKE / NOT LIKE

- The “LIKE” operator is used to select a column with a specific string pattern.
- The following wild cards is used with the *LIKE* operator.
  - %: nothing or anything (zero, one, or more characters)
  - _: a single character

```
SELECT employee_id, first_name, last_name
FROM employees
WHERE last_name LIKE '%s%e';
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME
1	98	Amber	Rose

```
SELECT employee_id, first_name, last_name
FROM employees
WHERE last_name LIKE '%r_';
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME
1	2	Jude	Rivera
2	7	Charles	Ward
3	6	Gabriel	Howard
4	16	Alex	Sanders
5	73	Lucy	Crawford
6	74	Elsie	Henry
7	43	Olivia	Ford
8	42	Amelia	Myers
9	25	Ronnie	Perry

Escape Character ‘\’

- To find the character patterns including ‘_’ (underscore) the escape character ‘\’ is used.

```
SELECT *
FROM class
WHERE course_code LIKE ‘DBS\_%’
```

- The above condition checks for course_code to starts ‘DBS_’ sub-string followed by zero, one, or more characters.

BETWEEN / NOT BETWEEN

```
SELECT product_id, product_name, list_price
FROM products
WHERE list_price BETWEEN 1000 AND 1050
```

	PRODUCT_ID	PRODUCT_NAME	LIST_PRICE
1	210	Intel Core i9-7900X	1029.99
2	54	Intel Xeon E5-1660 V3 (OEM/Tray)	1019.99
3	167	Intel Xeon E5-2650L V3 (OEM/Tray)	1010.46
4	214	Intel Core i7-5960X	1009.79
5	53	Intel Core 2 Extreme QX6800	1003.98

```
SELECT product_id, product_name, list_price
FROM products
WHERE list_price NOT BETWEEN 50 AND 8000
ORDER BY list_price;
```

	PRODUCT_ID	PRODUCT_NAME	LIST_PRICE
1	94	Western Digital WD2500AVVS	15.55
2	56	Western Digital WD2500AAJS	16.99
3	256	Western Digital WD5000AACS	26.99
4	235	Hitachi A7K1000-1000	41.99
5	168	Seagate ST31000340NS - FFP	43.99
6	268	Western Digital WD10EZEX	47.88
7	44	Seagate ST1000DM010	49.37
8	50	Intel SSDPECME040T401	8867.99

IN/ NOT IN

```
SELECT location_id, postal_code, city, state
FROM locations
WHERE city IN ('Munich', 'Toronto', 'Tokyo');
```

	LOCATION_ID	POSTAL_CODE	CITY	STATE
1	3	1689	Tokyo	Tokyo Prefecture
2	9	M5V 2L7	Toronto	Ontario
3	18	80925	Munich	Bavaria

```
SELECT *
FROM product_categories
WHERE category_id not in (2,3);
```

	CATEGORY_ID	CATEGORY_NAME
1	1	CPU
2	4	Mother Board
3	5	Storage

IS NULL / IS NOT NULL

```
SELECT location_id, postal_code, city, state
FROM locations
WHERE state IS NULL;
```

	LOCATION_ID	POSTAL_CODE	CITY	STATE
1	1	00989	Roma	(null)
2	2	10934	Venice	(null)
3	4	6823	Hiroshima	(null)
4	11	190518	Beijing	(null)
5	14	540198	Singapore	(null)
6	15	(null)	London	(null)

```
SELECT *
FROM warehouses
WHERE location_id IS NOT NULL;
```

	WAREHOUSE_ID	WAREHOUSE_NAME	LOCATION_ID
1	1	Southlake, Texas	5
2	2	San Francisco	6
3	3	New Jersey	7
4	4	Seattle, Washington	8
5	5	Toronto	9
6	6	Sydney	13
7	7	Mexico City	23
8	8	Beijing	11
9	9	Bombay	12

Logical Operators

Operator	Description
NOT	returns true if the condition is false
AND	returns true if all conditions are true
OR	returns true if either condition is true

AND / OR

```
SELECT product_id, product_name, list_price, category_id
FROM ass_products
WHERE category_id = 4 AND list_price < 280;
```

	PRODUCT_ID	PRODUCT_NAME	LIST_PRICE	CATEGORY_ID
1	180	Supermicro MBD-X10DAL-I-O	279.99	4

```
SELECT *
FROM ass_warehouses
WHERE warehouse_name = 'Bombay' OR location_id IN (6, 8, 13);
```

	WAREHOUSE_ID	WAREHOUSE_NAME	LOCATION_ID
1	2	San Francisco	6
2	4	Seattle, Washington	8
3	6	Sydney	13
4	9	Bombay	12

Rules of Precedence

Order of precedence	Operator
1	Parentheses
2	* /
3	+ -
4	=, <, >, <=, >=, <>
5	IS [NOT] NULL, [NOT] LIKE, [NOT] IN
6	BETWEEN
7	NOT
8	AND
9	OR

Concatenation Operator

- The concatenation operator links columns of type character strings.
  - Concatenation operator: ||
  - Concatenation function: CONCAT(column1, column2)

```
SELECT first_name, last_name,
       first_name || last_name as "Name",
       first_name || ' ' || last_name as "Full Name‟,
       CONCAT(first_name,last_name) as "Name 3",
       CONCAT(CONCAT(first_name, ' '), last_name) as "Name 3"
FROM contacts;
```

	FIRST_NAME	LAST_NAME	Name 1	Name 2	Name 3	Name 4
1	Flor	Stone	FlorStone	Flor Stone	FlorStone	Flor Stone
2	Lavera	Emerson	LaveraEmerson	Lavera Emerson	LaveraEmerson	Lavera Emerson
3	Fern	Head	FernHead	Fern Head	FernHead	Fern Head
4	Shyla	Ortiz	ShylaOrtiz	Shyla Ortiz	ShylaOrtiz	Shyla Ortiz
5	Jeni	Levy	JeniLevy	Jeni Levy	JeniLevy	Jeni Levy
6	Matthias	Hannah	MatthiasHannah	Matthias Hannah	MatthiasHannah	Matthias Hannah
7	Matthias	Cruise	MatthiasCruise	Matthias Cruise	MatthiasCruise	Matthias Cruise
8	Meenakshi	Mason	MeenakshiMason	Meenakshi Mason	MeenakshiMason	Meenakshi Mason

...

Literal Character Strings

- Literals such as a character, a number, or a date can be included in SQL select statement.
- Character and date literals must be enclosed by single quotations.

```
SELECT product_name || ' costs ' || list_price
from products;
```

	PRODUCT_NAME||'COSTS'||LIST_PRICE
1	Intel Xeon E5-2699 V3 (OEM/Tray) costs 3410.46
2	Intel Xeon E5-2697 V3 costs 2774.98
3	Intel Xeon E5-2698 V3 (OEM/Tray) costs 2660.72
4	Intel Xeon E5-2697 V4 costs 2554.99
5	Intel Xeon E5-2685 V3 (OEM/Tray) costs 2501.69
6	Intel Xeon E5-2695 V3 (OEM/Tray) costs 2431.95
7	Intel Xeon E5-2697 V2 costs 2377.09
8	Intel Xeon E5-2695 V4 costs 2269.99

...

```
SELECT last_name || q'['s job is ]' || job_title
FROM employees;
```

	LAST_NAME||Q'['SJOBIS]'||JOB_TITLE
1	Payne's job is Public Accountant
2	Stephens's job is Accounting Manager
3	Dunn's job is Administration Assistant
4	Bailey's job is President
5	Cooper's job is Administration Vice President
6	Rivera's job is Administration Vice President
7	Ramirez's job is Accountant
8	Gray's job is Accountant

...

Distinct

- Distinct clause is used to remove the duplicate rows from a SQL select query result.

```
SELECT DISTINCT state
FROM locations;
```

	STATE
1	(null)
2	Maharashtra
3	Distrito Federal,
4	Texas
5	Bavaria
6	New South Wales
7	BE
8	Geneve

...

Table Structure

- DESC or DESCRIBE
  - displays a table structure.

```
DESC products;
or
DESCRIBE products;
```

```
Name           Null     Type
-------------- -------- --------------
PRODUCT_ID     NOT NULL NUMBER
PRODUCT_NAME   NOT NULL VARCHAR2(255)
DESCRIPTION             VARCHAR2(2000)
STANDARD_COST           NUMBER(9,2)
LIST_PRICE              NUMBER(9,2)
CATEGORY_ID    NOT NULL NUMBER
```

Join

Displaying data from multiple tables

Fetching Data from Multiple Tables

- products

	PRODUCT_ID	PRODUCT_NAME	DESCRIPTION	STANDARD_COST	LIST_PRICE	CATEGORY_ID
1	228	Intel Xeon E5-2699 V3 (OEM/Tray)	Speed:2.3GHz,Cores:18,TDP:145W	2867.51	3410.46	1
2	248	Intel Xeon E5-2697 V3	Speed:2.6GHz,Cores:14,TDP:145W	2326.27	2774.98	1
3	249	Intel Xeon E5-2698 V3 (OEM/Tray)	Speed:2.3GHz,Cores:16,TDP:135W	2035.18	2660.72	1
4	2	Intel Xeon E5-2697 V4	Speed:2.3GHz,Cores:18,TDP:145W	2144.4	2554.99	1
5	45	Intel Xeon E5-2685 V3 (OEM/Tray)	Speed:2.6GHz,Cores:12,TDP:120W	2012.11	2501.69	1
6	46	Intel Xeon E5-2695 V3 (OEM/Tray)	Speed:2.3GHz,Cores:14,TDP:120W	1925.13	2431.95	1
7	47	Intel Xeon E5-2697 V2	Speed:2.7GHz,Cores:12,TDP:130W	2101.59	2377.09	1
8	51	Intel Xeon E5-2695 V4	Speed:2.1GHz,Cores:18,TDP:120W	1780.35	2269.99	1
9	91	Intel Xeon E5-2695 V2	Speed:2.4GHz,Cores:12,TDP:115W	1793.53	2259.99	1
10	92	Intel Xeon E5-2643 V2 (OEM/Tray)	Speed:3.5GHz,Cores:6,TDP:130W	1940.18	2200	1

...

- product_categories

	CATEGORY_ID	CATEGORY_NAME
1	1	CPU
2	2	Video Card
3	3	RAM
4	4	Mother Board
5	5	Storage

Joins

- What is the category name of each product?
  - To find the category name of each product, we need to select from two tables:
    - products
    - product_categories

- Different types of joins:
  - INNER JOIN (JOIN)
  - LEFT OUTER JOIN (LEFT JOIN)
  - RIGHT OUTER JOIN (RIGHT JOIN)
  - FULL OUTER JOIN (FULL JOIN)

INNER JOIN

- INNER JOIN returns all rows from multiple tables if the join condition is true.

```
SELECT p.product_id, p.product_name,
c.category_name
FROM ass_products p
INNER JOIN ass_product_categories c
ON p.category_id = c.category_id
ORDER BY p.product_id;
```

	PRODUCT_ID	PRODUCT_NAME	CATEGORY_NAME
1	1	G.Skill Ripjaws V Series	Storage
2	2	Intel Xeon E5-2697 V4	CPU
3	3	Corsair CB-9060011-WW	Video Card
4	4	AMD 100-505989	Video Card
5	5	PNY VCQK6000-PB	Video Card
6	6	Zotac ZT-P10810A-10P	Video Card
7	7	G.Skill Ripjaws V Series	Storage
8	8	Intel Xeon E5-1650 V4	CPU
9	9	Intel Xeon E5-2640 V4	CPU
10	10	Crucial	Storage

...

INNER JOIN (Example 2)

- What is the category name of products in category 1, 2, and 8.

```
SELECT p.product_id, p.product_name, c.category_name
FROM ass_products p
INNER JOIN ass_product_categories c
ON p.category_id = c.category_id
WHERE p.product_id in (1,2,8)
ORDER BY p.product_id;
```

	PRODUCT_ID	PRODUCT_NAME	CATEGORY_NAME
1	1	G.Skill Ripjaws V Series	Storage
2	2	Intel Xeon E5-2697 V4	CPU
3	8	Intel Xeon E5-1650 V4	CPU

SELF JOIN

- Who is the manager of employee with Id 101?
  - The employee Id is located in table employees.
  - The manager Id is also located in table employees.
  - A manager is also an employee. So, the column manager_id in table employees refers to the column employee_id in the same table.
- To find the manager name and last name of employee 101, we need to join the employee table with itself:

```
SELECT e.employee_id, e.first_name, e.last_name,
m.manager_id, m.first_name, m.last_name
FROM ass_employees e
INNER JOIN ass_employees m
ON e.manager_id = m.employee_id
WHERE e.employee_id = 101;
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	MANAGER_ID	FIRST_NAME_1	LAST_NAME_1
1	101	Annabelle	Dunn	1	Jude	Rivera

OUTER JOINs

- Display all customers and their orders. Include the customers without orders in your result.
- The **INNER JOIN** will select all rows from both tables as long as there is a match between the columns we are matching on.
- If a customer has not placed an order or has not placed an order in the time we might specify, then this customer will not be listed as there is no common field.
- To solve the problem, an outer join is required.
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN

LEFT / RIGHT OUTER JOIN

- LEFT (or RIGHT) JOIN returns
  - The result of the inner join between two tables (all matched rows)
  - And any unmatched rows from the left (or right) tables
- Display all customers and their orders. Include the customers without orders in your result.

```
SELECT c.customer_id, o.order_id, o.order_date
FROM ass_customers c
LEFT OUTER JOIN ass_orders o
ON c.customer_id = o.customer_id;
```

...

	CUSTOMER_ID	ORDER_ID	ORDER_DATE
100	47	97	12-JUL-16
101	48	98	18-MAR-17
102	49	99	07-JAN-17
103	16	100	05-JAN-17
104	17	103	08-FEB-16
105	18	104	01-FEB-17
106	306	(null)	(null)
107	136	(null)	(null)
108	86	(null)	(null)
109	104	(null)	(null)

...

FULL OUTER JOIN

- LEFT (or RIGHT) JOIN returns
  - The result of the inner join between two tables (all matched rows)
  - And any non-matching rows from both left and right tables

```
SELECT w.warehouse_id,
w.warehouse_name,
l.location_id
FROM ass_warehouses w
FULL OUTER JOIN ass_locations l
ON w.location_id = l.location_id;
```

	WAREHOUSE_ID	WAREHOUSE_NAME	LOCATION_ID	CITY
1	(null)	(null)	1	Roma
2	(null)	(null)	2	Venice
3	(null)	(null)	3	Tokyo
4	(null)	(null)	4	Hiroshima
5	1	Southlake, Texas	5	Southlake
6	2	San Francisco	6	South San Francisco
7	3	New Jersey	7	South Brunswick
8	4	Seattle, Washington	8	Seattle
9	5	Toronto	9	Toronto
10	(null)	(null)	10	Whitehorse
11	8	Beijing	11	Beijing
12	9	Bombay	12	Bombay
13	6	Sydney	13	Sydney
14	(null)	(null)	14	Singapore
15	(null)	(null)	15	London
16	(null)	(null)	16	Oxford
17	(null)	(null)	17	Stretford
18	(null)	(null)	18	Munich
19	(null)	(null)	19	Sao Paulo
20	(null)	(null)	20	Geneva
21	(null)	(null)	21	Bern
22	(null)	(null)	22	Utrecht
23	7	Mexico City	23	Mexico City
24	10	New Wharehouse	(null)	(null)