SQL

Select all from Book table

- SELECT * FROM Book

Order the tuples

- SELECT * FROM Member ORDER BY Name **<u>ASC</u>**
- SELECT * FROM Member ORDER BY Name **<u>DESC</u>**
- ASC – ascending | DESC - descending

Key words:

- AS '*something*' – rename attribute name
- DISTINCT(*attribute name*)– Only display attributes if the tuple is unique.
  Eg. DISTINCT('WenKang'), only displays the first tuple with the attribute name "WenKang".

Select all from Book table if ISBN = 1234

- SELECT * FROM Book WHERE ISBN = 1234
- Search Conditions
  - >, >=, <, <=
  - WHERE (condition1) **<u>OR</u>** (condition2)
  - WHERE (condition1) **<u>AND</u>** (condition2)
  - WHERE (attribute) **<u>BETWEEN</u>** 1000 AND 1500
  - WHERE (attribute) **<u>NOT BETWEEN</u>** 1000 AND 1500
  - WHERE (condition1) **<u>IN</u>** (1,3)
    * If value equals one of the numbers inside (does not mean BETWEEN)
  - WHERE (condition1) **<u>NOT IN</u>** (1,3)
  - WHERE (attribute) **<u>LIKE</u>** '*string*'
    * Used to search tuples that include string.
    * 'database' - database
    * '%database%' – 123database123
    * '%database' – 123database
    * 'database%' – database123

Scalar functions (String)

- LOWER(*column_name*) – Converts to lowercase
- HIGHER(*column_name*) – Converts to uppercase
- REPLACE(*column_name*) – Replaces string with other value
- STR(*column_name*) – Converts from number to string
- SUBSTRING(*column_name, start_index, length*) – Returns part of a string
  * start_index starts at 1. So "SQL" index 1 is "S"

Scalar functions (Math)

- CEILING() – Round up
- FLOOR() – Round down
- ROUND() – Round up to specified length or precision

Scalar functions (datetime)

- Date format = day/month/year
- GETDATE() – Returns current date
- DATENAME(year/day/month,datetime) – Return date as string
- DATEPART(year/day/month,datetime) – Return date as int
- DATEADD(year/day/month,number,datetime) – Add to date
- DATEDIFF(year/day/month,startdate,enddate) – Difference between 2 dates

Scalar functions (System)

Aggregate functions

- Count(attribute) – Returns the number of rows of the attribute
  * Ignores NULL values in column
  * Count(*) counts all columns
- MIN(attribute) – Returns the lowest value in a single column
- MAX(attribute) – Returns the highest value in a single column
- AVG(attribute) – Returns the average value of all values in a column
- SUM(attribute) – Returns the sum of all values in a column

JOIN function

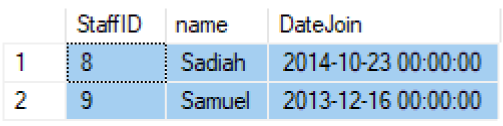
- Two Table Join

  SELECT * FROM tableName1 alias1
  INNER JOIN tableName2 alias2
  ON alias1.attribute = alias2.attribute

  ----

  * alias.attribute1 and alias.attribute2 must be primary and foreign key respectively.
  * Joins 2 tables together and **ON** align them to ensure that the primary and foreign are on the same tuple.
- INNER JOIN + WHERE
  SELECT * FROM tableName1 alias1 **INNER JOIN** tableName2 alias2 **ON** alias.attribute1 = alias.attribute2 WHERE condition

```
* SELECT s1.StaffID,s1.name,s1.DateJoin FROM Staff s1
INNER JOIN Staff s2 ON s1.SupervisorID = s2.StaffID
WHERE s2.Name = 'May May'
```



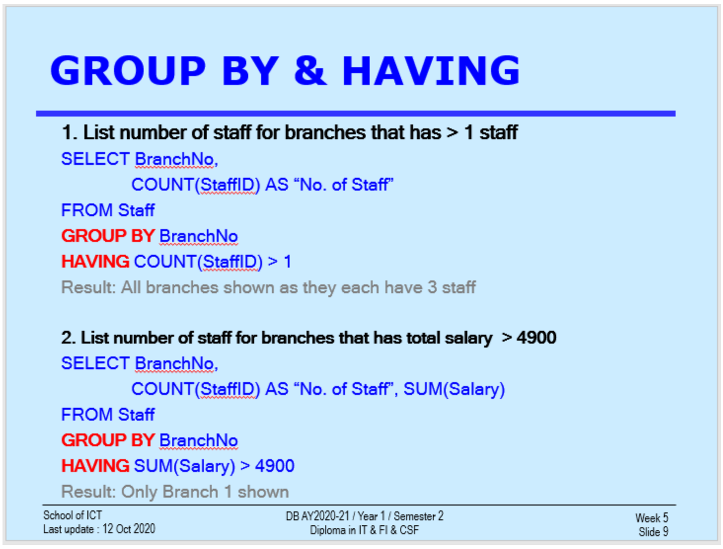
	StaffID	name	DateJoin
1	8	Sadiah	2014-10-23 00:00:00
2	9	Samuel	2013-12-16 00:00:00

- Three Table Join

  SELECT * FROM tableName1 alias1
  INNER JOIN tableName2 alias2
  ON alias1.attribute = alias2.attribute
  INNER JOIN tableName3 alias3
  ON alias1.attribute = alias2.attribute

GROUP BY

- Groups same attributes together into to sub-groups
- Selected attributes **must be** in the GROUP BY statement or aggregate function
- SELECT COUNT(attribute) FROM Staff GROUP BY BranchNo
- HAVING keyword
  * Specifies which group to **include** in result
  * Placed after GROUP BY statement



GROUP BY & HAVING

1. List number of staff for branches that has > 1 staff

```
SELECT BranchNo,
         COUNT(StaffID) AS "No. of Staff"
FROM Staff
GROUP BY BranchNo
HAVING COUNT(StaffID) > 1
```

Result: All branches shown as they each have 3 staff

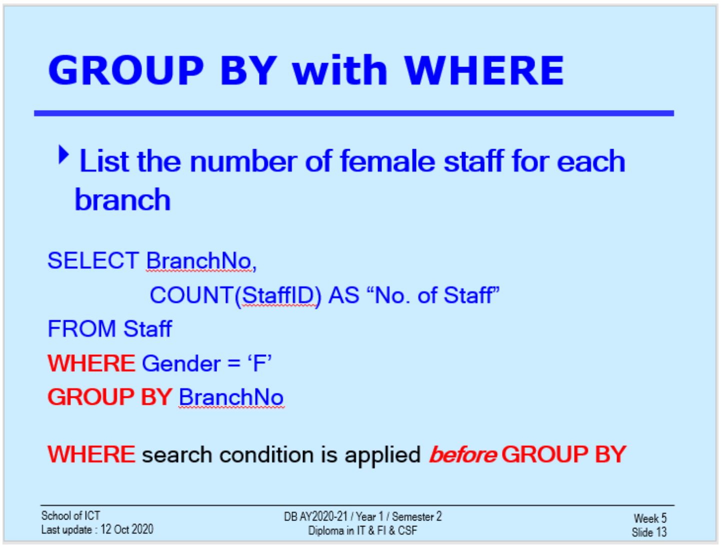
2. List number of staff for branches that has total salary > 4900

```
SELECT BranchNo,
         COUNT(StaffID) AS "No. of Staff", SUM(Salary)
FROM Staff
GROUP BY BranchNo
HAVING SUM(Salary) > 4900
```

Result: Only Branch 1 shown

School of ICT
Last update : 12 Oct 2020
DB AY2020-21 / Year 1 / Semester 2
Diploma in IT & FI & CSF
Week 5
Slide 9

- GROUP BY BranchNo, Gender
- WHERE keyword
  * Specifies which tuple to **include** in each group
  * Placed before GROUP BY statement



GROUP BY with WHERE

- List the number of female staff for each branch

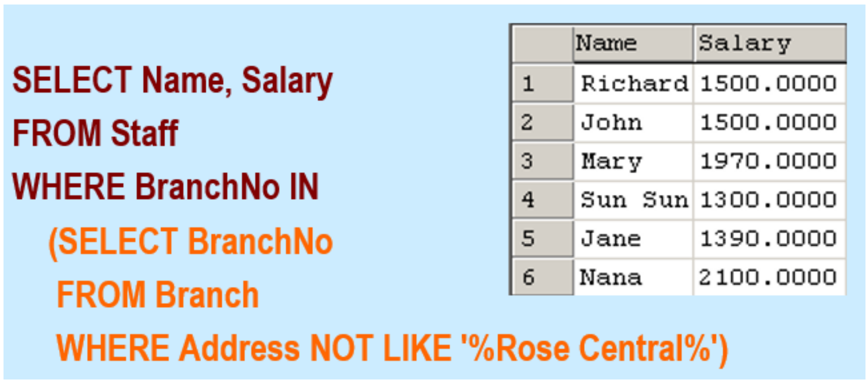
```
SELECT BranchNo,
         COUNT(StaffID) AS "No. of Staff"
FROM Staff
WHERE Gender = 'F'
GROUP BY BranchNo
```

WHERE search condition is applied *before* GROUP BY

School of ICT
Last update : 12 Oct 2020
DB AY2020-21 / Year 1 / Semester 2
Diploma in IT & FI & CSF
Week 5
Slide 13

Subquery

- Query inside a query
  * ORDER BY is only allowed in the outer query



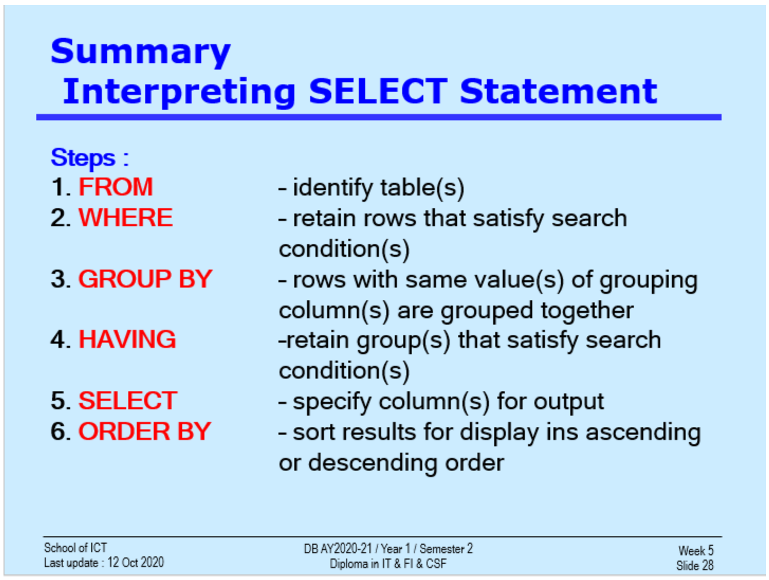
```
SELECT Name, Salary
FROM Staff
WHERE BranchNo IN
    (SELECT BranchNo
     FROM Branch
     WHERE Address NOT LIKE '%Rose Central%')
```

	Name	Salary
1	Richard	1500.0000
2	John	1500.0000
3	Mary	1970.0000
4	Sun Sun	1300.0000
5	Jane	1390.0000
6	Nana	2100.0000

JOIN vs Subquery

- JOIN
  * When data from both tables are needed
- Subquery
  * Comparison with an aggregate function

SELECT summery



Summary Interpreting SELECT Statement

Steps :

1. FROM	- identify table(s)
2. WHERE	- retain rows that satisfy search condition(s)
3. GROUP BY	- rows with same value(s) of grouping column(s) are grouped together
4. HAVING	-retain group(s) that satisfy search condition(s)
5. SELECT	- specify column(s) for output
6. ORDER BY	- sort results for display ins ascending or descending order

School of ICT
Last update : 12 Oct 2020
DB AY2020-21 / Year 1 / Semester 2
Diploma in IT & FI & CSF
Week 5
Slide 28

Creating tables:

** Note:

- **CANNOT** reference other columns in *column_constraint*

CREATE TABLE *tableName*{

column_name format null/not null [*column_constraint*],

column_name format null/not null [*column_constraint*],

column_name format null/not null [*column_constraint*],

CONSTRAINT *name* PRIMARY KEY (*column_name*),

CONSTRAINT *name* FOREIGN KEY REFERENCES *table_name*(*column_name*),

CONSTRAINT *name* FOREIGN KEY REFERENCES *table_name*(*column_name*),

CONSTRAINT *name* CHECK (*condition*)

}

```
CREATE TABLE Member{
MemberID int not null,
Name varchar(50) NOT NULL,
Address varchar(150) NULL,
BranchNo int NOT NULL,
Gender char(1) NOT NULL CHECK (Gender IN (‘M’,’F’)),
CONSTRAINT PK_Member PRIMARY KEY (MemberID),
CONSTRAINT FK_Member FOREIGN KEY REFERENCES Branch(BranchNo),
//--Constraint Reservation_ED CHECK (EndDate >= ResDate),
}
```

Delete Table:

DROP TABLE *table_name*

Inserting values:

* When dealing with a foreign key, ensure that the primary key in another table is there.

INSERT INTO *table_name* VALUES (DEFAULT/NULL/*value*)

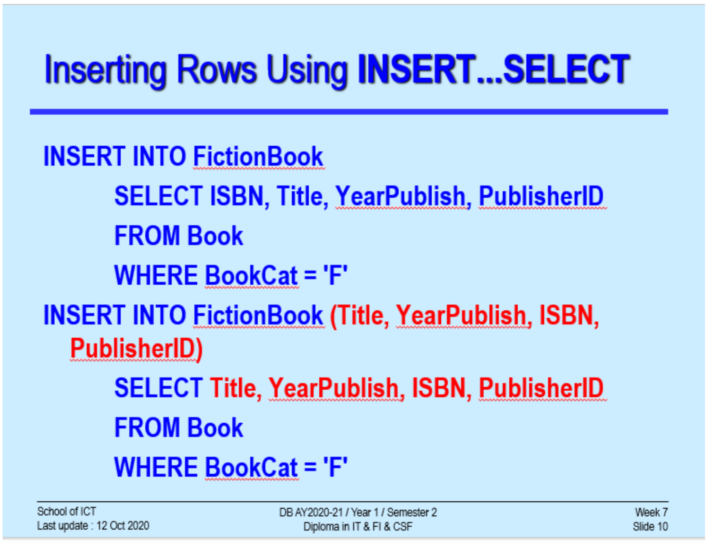
- `INSERT INTO Book (ISBN,Title,YearPublish,PublisherID,BookCat)`
  `VALUES (‘01020310’,’In your hands.’,1975,6,’NF’)`

INSERT INTO *table_name* (column_list) VALUES (DEFAULT/NULL/*value*)

- INSERT INTO Book
  `VALUES (‘01020310’,’In your hands.’,1975,6,’NF’)`

INSERT INTO *table_name* SELECT (column_list) FROM table_name WHERE (condition)

INSERT INTO *table_name* (*column_list*) SELECT (*column_list*) FROM *table_name* WHERE (*condition*)



Inserting Rows Using INSERT...SELECT

```
INSERT INTO FictionBook
        SELECT ISBN, Title, YearPublish, PublisherID
        FROM Book
        WHERE BookCat = 'F'
INSERT INTO FictionBook (Title, YearPublish, ISBN,
   PublisherID)
        SELECT Title, YearPublish, ISBN, PublisherID
        FROM Book
        WHERE BookCat = 'F'
```

School of ICT
Last update : 12 Oct 2020

DB AY2020-21 / Year 1 / Semester 2
Diploma in IT & FI & CSF

Week 7
Slide 10

Deleting values:

Delete all rows => DELETE *table_name*

Delete specific rows

- ➩ DELETE *table_name* WHERE (*condition*)
- ➩ E.g. DELETE Member WHERE Name = 'Tan Mei Ling'

DELETE table_name FROM table_name ()

Updating values:

new_value can be DEFAULT/NULL or value

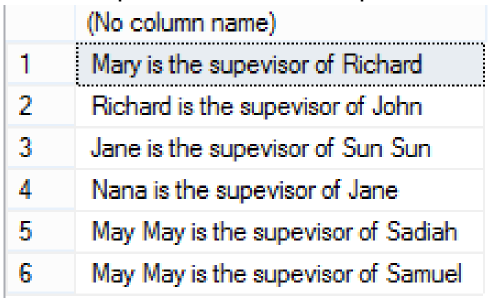
UPDATE *table_name* SET *column_name*= *new_value* WHERE (*condition*)

- ➩ UPDATE Branch SET Address ='Tile 32' WHERE Address = '%street 32%'

UPDATE *table_name* SET *column_name*= *new_value, column_name*= *new_value*, WHERE (*condition*)

Concatenating strings

- CONCAT (*string1*,*string2* …)
- ➩
```
SELECT CONCAT(sup.name, ' is the supevisor of ', s.name)
FROM staff s INNER JOIN staff sup
ON s.SupervisorID = sup.StaffID
```



	(No column name)
1	Mary is the supevisor of Richard
2	Richard is the supevisor of John
3	Jane is the supevisor of Sun Sun
4	Nana is the supevisor of Jane
5	May May is the supevisor of Sadiah
6	May May is the supevisor of Samuel