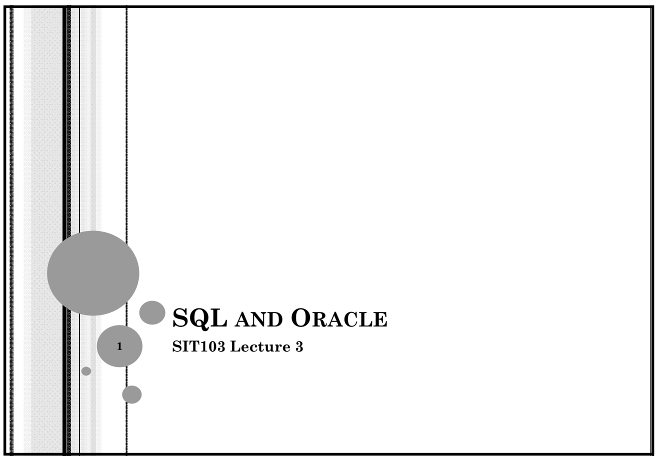
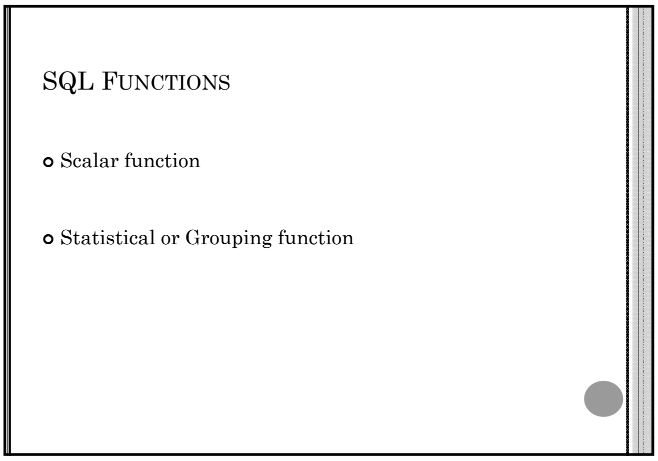


SQL and Oracle

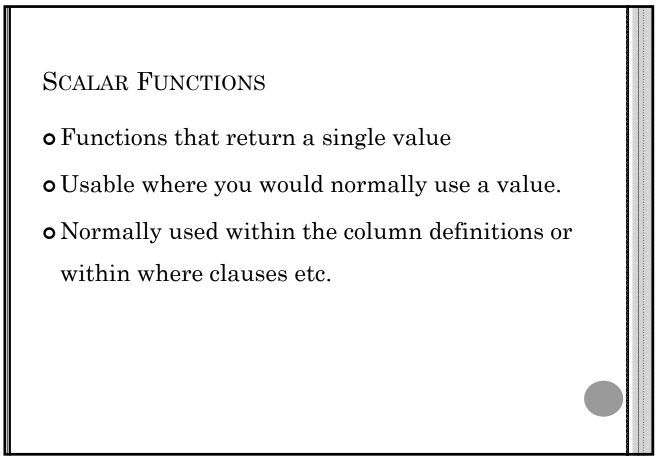
SIT103 Lecture 3

1



SQL Functions

- Scalar function
- Statistical or Grouping function



Scalar Functions

- Functions that return a single value
- Usable where you would normally use a value.
- Normally used within the column definitions or within where clauses etc.

SCALAR FUNCTIONS

- DECODE(TARGET,VAL1,RES1,VAL2,RES2...)
  - Eg: DECODE(SEX, 'M','Male','F','Female')
- FLOOR
  - Remove the fraction part of a number
- INITCAP
  - Make first letter upper and rest lower
- LOWER
  - Lower case all characters.
- LTRIM
  - Left Trim. Remove leading spaces.

SCALAR FUNCTIONS

- NVL(TARGET,REPLACEVAL)
  - Null value. Replace any nulls with a message.
- RTRIM
  - Right Trim. Remove trailing spaces.
- SOUNDEX
  - Creates a code for the way the value sounds.
- UPPER
  - Upper Case all characters.

FORMATTING DATES

- TO_CHAR Function used to display dates.
  TO_CHAR(FIELD,FORMAT)
- The format string can be made up of many formatting codes that you can look up on the on-line documentation (Format Models in TO_CHAR)
- Examples:

```
SELECT  TO_CHAR(DOB,'DD/MM/YY')
SELECT  TO_CHAR(DOB,'DD Month, YYYY')
SELECT  TO_CHAR(DOB,'DAY')
SELECT  TO_CHAR(DOB,'CC')||' Century'
```

Statistical Functions

- There are five basic statistical functions.
- They are also known as “grouping” functions.
- All statistical functions return one value only, no matter how many rows they operate on.
- When they are used, values of individual rows cannot be displayed.
- Can be used with usual where clauses.

Statistical Functions - COUNT

- COUNT
  - Counts number of rows

```
SELECT COUNT(*)
FROM COURSE;
```

```
COUNT(*)
---------
     4469
```

```
SELECT COUNT(*)
FROM STUDENT
WHERE SURNAME =
  'SMITH' ;
```

```
COUNT(*)
---------
        2
```

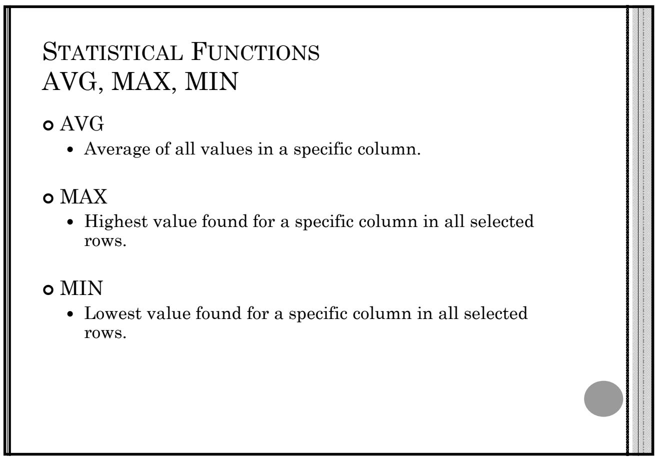
Statistical Functions - SUM

- SUM
  - Add up values in a specified column for all selected rows.

```
SELECT SUM(FEE)
FROM COURSE
WHERE DEPTNO = 100 ;
```

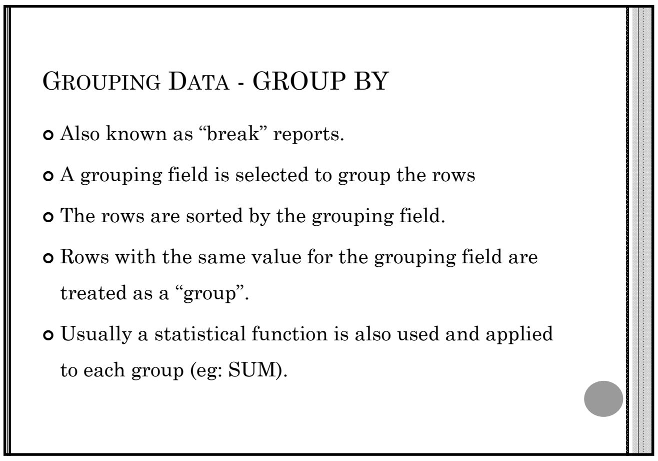
```
   SUM(FEE)
-----------

$100,234.00
```

- Result would be a single value of all the fees in the selected rows added up.



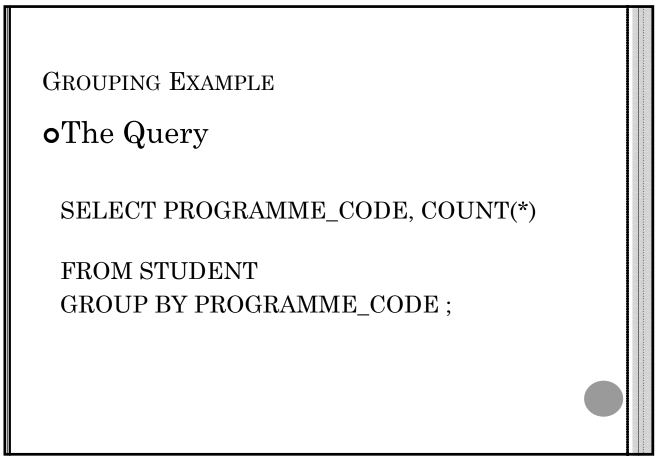
Statistical Functions AVG, MAX, MIN

- AVG
  - Average of all values in a specific column.
- MAX
  - Highest value found for a specific column in all selected rows.
- MIN
  - Lowest value found for a specific column in all selected rows.



Grouping Data - GROUP BY

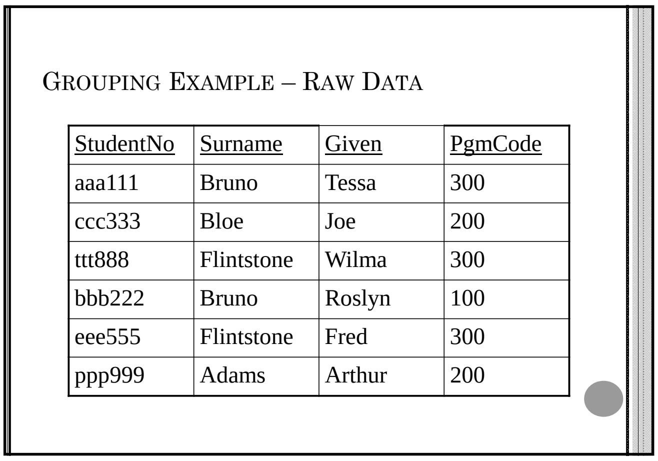
- Also known as “break” reports.
- A grouping field is selected to group the rows
- The rows are sorted by the grouping field.
- Rows with the same value for the grouping field are treated as a “group”.
- Usually a statistical function is also used and applied to each group (eg: SUM).



Grouping Example

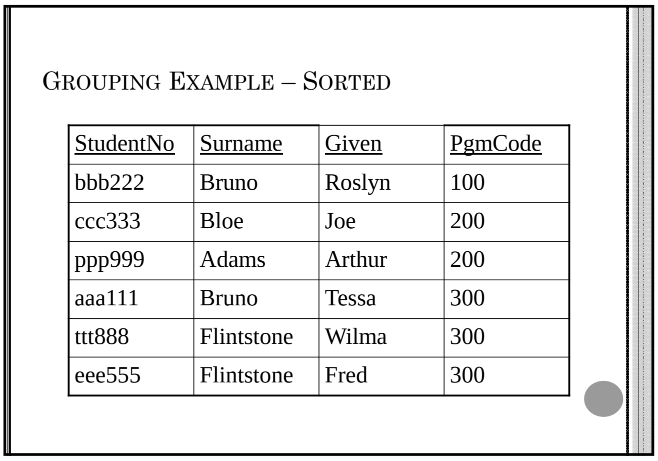
- The Query

```
SELECT PROGRAMME_CODE, COUNT(*)

FROM STUDENT
GROUP BY PROGRAMME_CODE ;
```



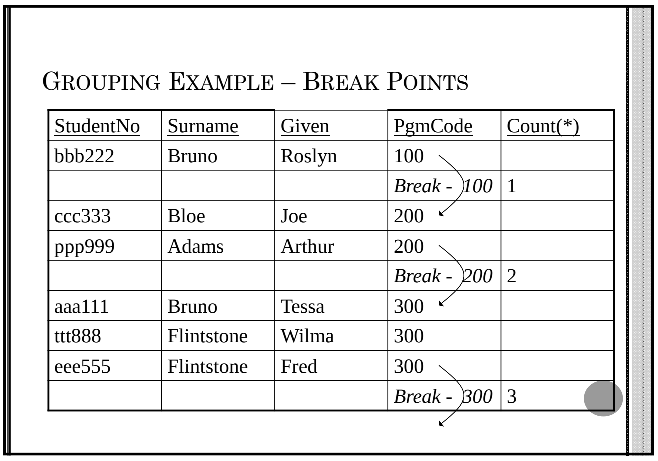
Grouping Example – Raw Data

StudentNo	Surname	Given	PgmCode
aaa111	Bruno	Tessa	300
ccc333	Bloe	Joe	200
ttt888	Flintstone	Wilma	300
bbb222	Bruno	Roslyn	100
eee555	Flintstone	Fred	300
ppp999	Adams	Arthur	200



Grouping Example – Sorted

StudentNo	Surname	Given	PgmCode
bbb222	Bruno	Roslyn	100
ccc333	Bloe	Joe	200
ppp999	Adams	Arthur	200
aaa111	Bruno	Tessa	300
ttt888	Flintstone	Wilma	300
eee555	Flintstone	Fred	300



Grouping Example – Break Points

StudentNo	Surname	Given	PgmCode	Count(*)
bbb222	Bruno	Roslyn	100	
			Break - 100	1
ccc333	Bloe	Joe	200	
ppp999	Adams	Arthur	200	
			Break - 200	2
aaa111	Bruno	Tessa	300	
ttt888	Flintstone	Wilma	300	
eee555	Flintstone	Fred	300	
			Break - 300	3

Grouping Example – Result

PgmCode	Count(*)
100	1
200	2
300	3

Grouping Data – HAVING

- The HAVING clause operates like a WHERE clause, but is applied to the grouping value.
- WHERE is applied to each row before the grouping operation is done.
- HAVING is applied after the grouping is performed and operates on the calculated grouping value (before it is displayed).

Grouping Data – HAVING

```
SELECT PROGRAMME_CODE, COUNT(*)
FROM STUDENT
GROUP BY PROGRAMME_CODE
HAVING   COUNT(*) >= 2 ;
```

PgmCode	Count(*)
100	1
200	2
300	3

These are selected and displayed

Defining Tables

```
      CREATE TABLE STUDENT
            ( STUDENT_NO              CHAR(8)         NOT
       NULL,
              SURNAME            VARCHAR(30),
              GIVEN              VARCHAR(30),
              DOB                DATE,
              .............etc   );

     CREATE TABLE COURSE
            ( COURSE_CODE        CHAR(5)     PRIMARY KEY,
              COURSE_NAME        VARCHAR(50),
              DEPT_NO            NUMERIC(5),
              FEE                NUMERIC(15,2));
```

Oracle Field Types

- Oracle specific native types
  - CHAR(<length>)
  - VARCHAR2(<length>)
  - DATE
  - NUMBER(<precision>,<scale>)

ANSI Field Types

- Set of types defined as a standard.
- Oracle maps these to native types
  - CHAR(<length>)
  - VARCHAR(<length>)
  - DATE
  - NUMERIC(<precision>,<scale>)
  - DECIMAL (<precision>,<scale>),
  - INT, FLOAT

Working with Tables

- Deleting a Table

```
DROP TABLE STUDENT;
```

- Copying a table
  - Creates a brand new table.

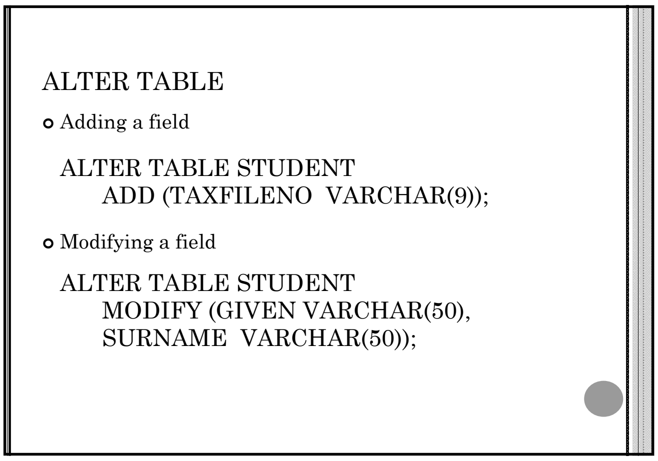
```
CREATE TABLE SMITHSTUD AS
   SELECT *
   FROM STUDENT
   WHERE SURNAME = 'SMITH';
```

Modifying Table Structures

- Some elements of a table structure can be altered after creation using ALTER TABLE.
- Each database has different rules about what can be altered so the ALTER TABLE command differs on each platform.
- The abilities also change over time for a given database, so you often need to check the latest documentation.

ALTER TABLE

- Oracle allows:
  - Adding or Dropping a column
  - Changing the type of a column (if values permit)
  - Enlarging the length of a column
  - Reducing the length of a column (if table empty)
  - Adding, modifying and dropping constraints
  - Renaming a table
  - Many others
    - look up the ALTER TABLE command



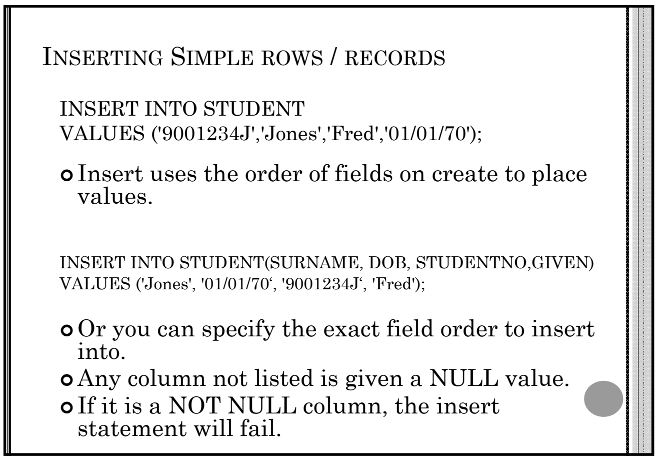
ALTER TABLE

- Adding a field

```
ALTER TABLE STUDENT
    ADD (TAXFILENO  VARCHAR(9));
```

- Modifying a field

```
ALTER TABLE STUDENT
    MODIFY (GIVEN VARCHAR(50),
    SURNAME  VARCHAR(50));
```



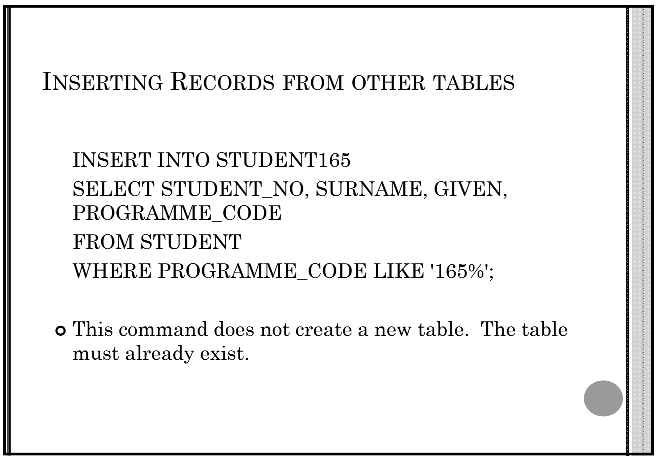
INSERTING SIMPLE ROWS / RECORDS

```
INSERT INTO STUDENT
VALUES ('9001234J','Jones','Fred','01/01/70');
```

- Insert uses the order of fields on create to place values.

```
INSERT INTO STUDENT(SURNAME, DOB, STUDENTNO,GIVEN)
VALUES ('Jones', '01/01/70', '9001234J', 'Fred');
```

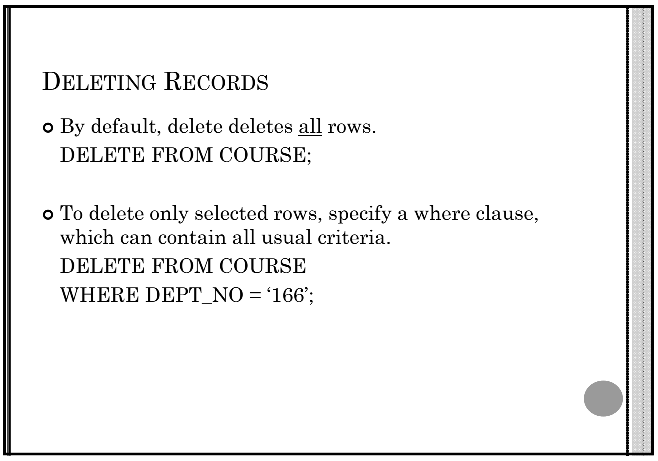
- Or you can specify the exact field order to insert into.
- Any column not listed is given a NULL value.
- If it is a NOT NULL column, the insert statement will fail.



INSERTING RECORDS FROM OTHER TABLES

```
INSERT INTO STUDENT165
SELECT STUDENT_NO, SURNAME, GIVEN,
PROGRAMME_CODE
FROM STUDENT
WHERE PROGRAMME_CODE LIKE '165%';
```

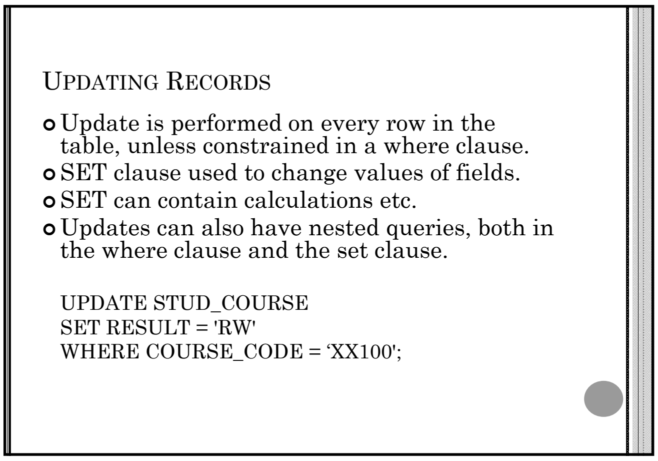
- This command does not create a new table. The table must already exist.



Deleting Records

- By default, delete deletes <u>all</u> rows.
  DELETE FROM COURSE;

- To delete only selected rows, specify a where clause, which can contain all usual criteria.
  DELETE FROM COURSE
  WHERE DEPT_NO = '166';



Updating Records

- Update is performed on every row in the table, unless constrained in a where clause.
- SET clause used to change values of fields.
- SET can contain calculations etc.
- Updates can also have nested queries, both in the where clause and the set clause.

```
UPDATE STUD_COURSE
SET RESULT = 'RW'
WHERE COURSE_CODE = 'XX100';
```