SQL: Data Manipulation Language (DML) Part 1

SCSD2523 Database

Semester 1 2019/2020

Learning Objective

- At the end of the topic, students will be able to:
  - Write DML statements to add data into table, update data, and delete data using following commands:
    - **INSERT INTO … VALUES …**
    - **INSERT INTO … SELECT …**
    - **UPDATE … SET … [WHERE] …**
    - **DELETE FROM … [WHERE] …**
  - Write DML statement to display records in tables
    - **SELECT … FROM …**
    - **SELECT … FROM … WHERE**
    - **SELECT … FROM … WHERE … BETWEEN … AND**
    - **SELECT … FROM … WHERE … IN**
    - **SELECT … FROM … WHERE … LIKE**

SQL Statements

STATEMENTS	TYPE
SELECT INSERT UPDATE DELETE MERGE	**DATA MANIPULATION LANGUAGE (DML)** Retrieves data from database, enters new rows, changes existing rows, and removes unwanted rows from tables in the database, respectively
CREATE ALTER DROP RENAME TRUNCATE COMMENT	**DATA DEFINITION LANGUAGE (DDL)** Sets up, changes, and removes data structures from tables
GRANT REVOKE	**DATA CONTROL LANGUAGE (DCL)** Provides or removes access rights to both the Database and the structures within it
COMMIT ROLLBACK SAVEPOINT	**TRANSACTION CONTROL** Manages the changes made by DML statements. Changes to the data can be grouped together into logical transactions

Writing SQL Statements

- SQL statements are **not case sensitive** (unless indicated).
- SQL statements can be entered on **one or more lines.**
- Keywords cannot be abbreviated or split across lines
- Clauses are usually placed on separate lines
- **Indents** are used to enhance readability

Human Resource (HR) schema

- In the HR records, each employee has an identification number, email address, job identification code, salary, and manager. Some employees earn commissions in addition to their salary
- The company also tracks information about jobs within the organization. Each job has an identification code, job title, and a minimum and maximum salary range for the job. Some employees have been with the company for a long time and have held different positions within the company. When an employee resigns, the duration the employee was working for, the job identification number, and the department are recorded.

Human Resource (HR) schema

- The sample company is regionally diverse, so it tracks the locations of its warehouses and departments. Each employee is assigned to a department, and each department is identified by a unique department number or short name. Each department is associated with one location, and each location has a full address that includes the street name, postal code, city, state or province, and the country code.
- In places where the departments and warehouses are located, the company records details such as the country name, currency symbol, currency name, and the region where the country is located geographically.

INSERT INTO … VALUES

- Purpose: To **insert a row of data** into a table.
- Syntax:

```
INSERT INTO tableName (column1, column2, column3)
VALUES (value1, value2, value3);
```

- Example: Insert a new row into table Department:

```
INSERT INTO departments
VALUES (70,'Public Relations',100,70);
```

OR

```
INSERT INTO departments (department_id, department_name,
manager_id, location_id)
VALUES (70,'Public Relations',100,70);
```

NOTE: Use single quote (') for string / character data

Inserting rows with NULL values

- Implicit method: Omit the column from the column list

```
INSERT INTO departments (department_id, department_name)
VALUES (30, 'Purchasing');
```

- Explicit method: Specify the NULL keyword in the VALUES clause

```
INSERT INTO departments (department_id, department_name)
VALUES (30, 'Purchasing', NULL, NULL);
```

department_id	department_name	manager_id	location_id
30	Purchasing		

Attention: Inserting NULL values

- Be sure that you can use the **NULL** value in the targeted column by verifying the NULL status with the **DESCRIBE** command

Result Grid | Filter Rows: | Export: | Wra

Field	Type	Null	Key	Default	Extra
Dept_Id	char(1)	NO	PRI	NULL	
Dept_Name	varchar(20)	YES		NULL	
Manager_Id	int(11)	YES		NULL	
location	varchar(100)	NO		NULL	

Output of DESCRIBE in MySQL

Script Output | Task completed in 0.055 seconds

```
Name          Null?    Type
------------- -------- -------------
DEPT_ID       NOT NULL CHAR(1)
DEPT_NAME              VARCHAR2(20)
MANAGER_ID             NUMBER(38)
LOCATION      NOT NULL VARCHAR2(100)
```

Output of DESCRIBE in Oracle

Attention: Inserting NULL values

- Common errors that can occur during the user input are checked in the following order:
  - Mandatory value missing for a **NOT NULL** column
  - Duplicate value violating any **unique** or **primary key** constraint
  - Any value violating a **CHECK** constraint
  - **Referential integrity maintained** for a foreign key constraint
  - Data **type mismatch** or values too wide to fit in column

Attention: Inserting NULL values

- Any value violating a CHECK constraint
  - CHECK is a constraint that set a specific condition for the input data to follow. E.g.: DDL statement below:

```
/* Create the table "Dept" with constraints
create table Dept (
Dept_Id Char(1),
Dept_Name Varchar(20),
Manager_Id integer check (Manager_Id>0),
location Varchar(100) not null,
Constraint dept_pk Primary Key (Dept_Id));
```

BUT!!
CHECK constraint doesn't work in MySQL. You can insert the constraint but MySQL will not execute it.

  - Example above: the column "Manager_Id" has a CHECK constraint that checks if the input is larger than zero,
  - Therefore only input that is larger than zero is allowed to enter into the column. Error example in Oracle:

```
Error starting at line : 32 in command -
INSERT INTO Dept (Dept_Id, Dept_Name, Manager_Id, location) VALUES ('B', 'Accounting', 0, 'KL')
Error report -
ORA-02290: check constraint (SYSTEM.SYS_C007120) violated
```

Attention: Inserting NULL values

- **Referential integrity maintained** for a foreign key constraint
  - Whenever the foreign key exists in the table, according to **Referential Integrity constraint**, the value can either be:
    - **Values exist** in the parent table foreign key refers to. OR
    - Wholly **NULL**
  - Therefore, non-NULL values that does not exist at the table where the foreign key refers to are not allowed to enter in the column. Error example in Oracle:

```
Error starting at line : 34 in command -
INSERT INTO Staff(Emp_Id, Emp_Name, Emp_Date, Post_Id, Dept_Id) VALUES (101, 'Alan', '10-10-2018',102,'A')
Error report -
ORA-02291: integrity constraint (SYSTEM.SYS_C007123) violated - parent key not found
```

Recommendation: Inserting NULL values

- Use of the column list is **recommended** because it makes the INSERT statement more readable and reliable, or less prone to mistakes. Example

```
INSERT INTO tableName (column1, column2, column3)
VALUES (value1, value2, value3);
```

Inserting Special Values

- The SYSDATE function records the current date and time

```
INSERT INTO departments (employee_id, first_name,
last_name, email, phone_number, hire_date, job_id,
salary, commission_pct, manager_id, department_id)
VALUES (113, 'Harraz', 'Ahmad Faizal', 'HAHMMADFAIZAL',
'07-5531234', SYSDATE, 'AC_ACCOUNT', 7000, NULL,
205,110);
```

Inserting specific Date and Time

- The DD-MON-RR format is generally used to insert a date value.
- You may also supply the date value in DD-MON-YYYY format.
- This is recommended because it clearly specifies the century and does not depend on the internal RR format logic specifying the correct century

```
INSERT INTO employees (employee_id, first_name,
last_name, email, phone_number, hire_date, job_id,
salary, commission_pct, manager_id, department_id)
VALUES (114, 'Yusuf', 'Syarin', 'YSYARIN', '07-5534567',
'09-NOV-2016', 'SA_REP', 8000, 0.2, 100, 60);
```

For MySQL, use "YYYY-MM-DD" format for DATE datatype

INSERT INTO … SELECT

- To insert data **from an existing table**
- Write the INSERT statement with a **subquery**
- DO NOT use the VALUES clause
- Match the number of columns in the INSERT clause to those in the subquery

```
INSERT INTO sales_reps (id, name, salary, commission_pct)
SELECT employee_id, last_name, salary, commission_pct
FROM employees
WHERE job_id LIKE '%REP%';
```

UPDATE … SET

- Purpose: To modify existing values in a table.
- Syntax:

[ ] = OPTIONAL

```
UPDATE table-name
SET col1 = update-value[,col2 = update-value]
[WHERE search-condition]
```

- Example: Update the department ID of employee with the ID of 113 to 50

```
UPDATE employees
SET department_id = 50
WHERE employee_id = 113;
```

Update Two Columns with a Subquery

- Can also be done for multiple subqueries

[ ] = OPTIONAL

```
UPDATE      table
SET         column = (SELECT column
            FROM table
            WHERE condition)
            [,column2 = (SELECT column
            FROM table
            WHERE condition)]
[WHERE condition]
```

Update Two Columns with Subquery

- Example: Update employee 113's job and salary to match those of employee 205

```
UPDATE employees
SET (job_id, salary) = (SELECT job_id, salary FROM
employees WHERE employee_id = 205)
WHERE employee_id = 113;
```

DELETE FROM

- Purpose: To delete existing rows from table
- Syntax:

[ ] = OPTIONAL

```
DELETE FROM table-name
[WHERE search-condition]
```

- Example: Delete the record of department Finance

```
DELETE FROM departments
WHERE department_name = 'Finance';
```

- All rows in the table are deleted if you omit the WHERE clause.

```
DELETE FROM departments
```

SELECT … FROM

- To retrieve and display data from one or more tables.

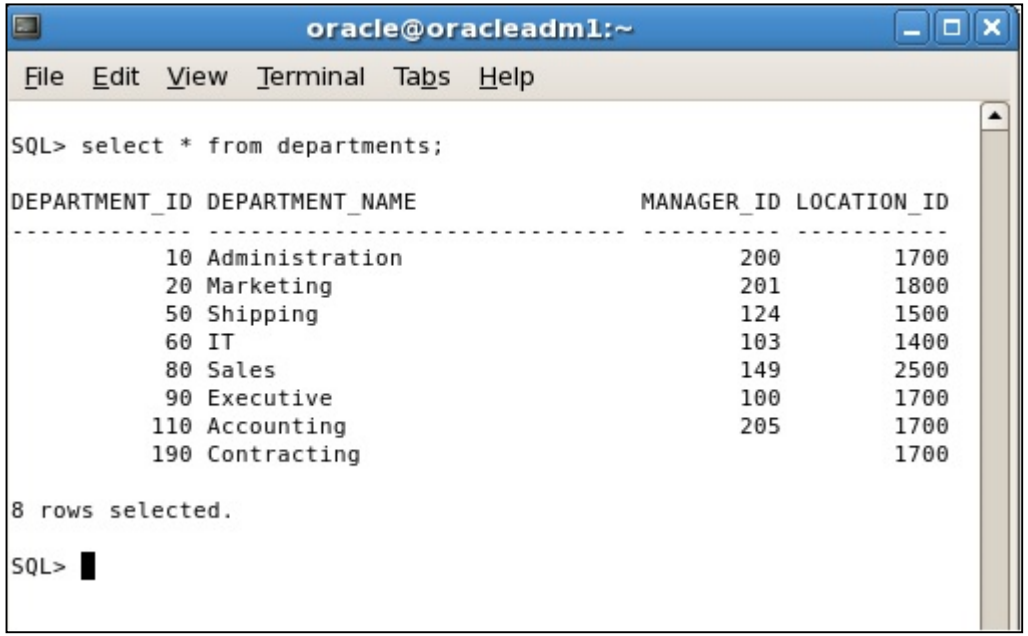
[ ] = OPTIONAL

```
SELECT col1, col2, … coln
FROM TableName [,TableName]
[WHERE condition]
[GROUP BY columnList]
[HAVING condition]
[ORDER BY columnList]
```

SELECT … FROM

- Retrieve all columns and all rows
- List the full details of the DEPARTMENTS table

```
SELECT * FROM departments;
```



Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

SELECT … FROM

- Retrieve specific columns, all rows
- Example:
- Produce a list of salaries for all staff, displaying only the employee ID, first name and salary

```
SELECT employee_id,
first_name, salary
FROM departments;
```

```
SQL> select employee_id, first_name, salary
  2  from employees;

EMPLOYEE_ID FIRST_NAME               SALARY
----------- -------------------- ----------
        200 Jennifer                   4400
        201 Michael                   13000
        202 Pat                        6000
        205 Shelley                   12000
        206 William                    8300
        100 Steven                    24000
        101 Neena                     17000
        102 Lex                       17000
        103 Alexander                  9000
        104 Bruce                      6000
        107 Diana                      4200

EMPLOYEE_ID FIRST_NAME               SALARY
----------- -------------------- ----------
        124 Kevin                      5800
        141 Trenna                     3500
        142 Curtis                     3100
        143 Randall                    2600
        144 Peter                      2500
        149 Eleni                     10500
        174 Ellen                     11000
        176 Jonathon                   8600
        178 Kimberely                  7000

20 rows selected.
```

INSERT INTO … SELECT

- Copy multiple row of records into a different table
- Example
  - Create a new table name MYDEPT with the same structure as table DEPT

```
CREATE TABLE MYDEPT (DEPTNO, INT NOT
NULL, DNAME   VARCHAR(20), LOC
VARCHAR(20));
```

  - Copy all records from DEPT table to MYDEPT

```
INSERT INTO MYDEPT
(SELECT * FROM DEPT);
```

```
SQL> create table mydept (deptno int not null, dname

Table created.

SQL> select * from dept
  2  ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> insert into mydept
  2  (select * from dept);

4 rows created.

SQL> select * from mydept;

    DEPTNO DNAME                LOC
---------- -------------------- --------------------
        10 ACCOUNTING           NEW YORK
        20 RESEARCH             DALLAS
        30 SALES                CHICAGO
        40 OPERATIONS           BOSTON
```

Create new table by copying the structure of an existing table

- Create a new table name DEPT_BARU with the same structure as table DEPARTMENTS.

```
CREATE TABLE dept_baru
AS (SELECT * FROM departments
WHERE 1=2);
```

```
SQL> create table dept_baru
  2  as (select *
  3      from departments
  4      where 1=2);

Table created.

SQL> select *
  2  from dept_baru;

no rows selected
```

- This would create a new table called DEPT_BARU that includes all columns from the DEPARTMENTS table WITHOUT the data from DEPARTMENTS

Create new table by copying the structure of an existing table (and data)

- Create a new table name NEW_DEPT with the same structure as table DEPARTMENTS.

```
CREATE TABLE new_dept
AS (SELECT * FROM departments);
```

```
SQL> create table new_dept
  2  as (select *
  3      from departments);

Table created.

SQL> select * from new_dept;

DEPARTMENT_ID DEPARTMENT_NAME                MANAGER_ID LOCATION_ID
------------- ------------------------------ ---------- -----------
           10 Administration                        200        1700
           20 Marketing                             201        1800
           50 Shipping                              124        1500
           60 IT                                    103        1400
           80 Sales                                 149        2500
           90 Executive                             100        1700
          110 Accounting                            205        1700
          190 Contracting                                      1700

8 rows selected.
```

- This would create a new table called NEW_DEPT that includes all columns from the DEPARTMENTS table INCLUDING all the data from DEPARTMENTS

SELECT…FROM…WHERE (1)

- To list only selected row of records based on conditions
- Conditions involve comparison operators and/or logical operators
  - Comparison operators:
    - = <> < > <= >= !=
    - BETWEEN..AND LIKE IN(set)
  - Logical operators:
    - AND OR NOT

```
SELECT list-of-columns
FROM list of tables
WHERE search-condition;
```

SELECT…FROM…WHERE…BETWEEN…AND

- Example:
  - List all employee ID, last_name, salary where salary is between 2000 and 5000

```
SELECT employee_id, last_name, salary
FROM employees
WHERE salary
BETWEEN 2000 AND 5000;
```

```
SQL> select employee_id, last_name, salary
  2  from employees
  3  where salary between 2000 and 5000;

EMPLOYEE_ID LAST_NAME                     SALARY
----------- ------------------------- ----------
        200 Whalen                          4400
        107 Lorentz                         4200
        141 Rajs                            3500
        142 Davies                          3100
        143 Matos                           2600
        144 Vargas                          2500

6 rows selected.
```

SELECT…FROM…WHERE…IN

- To search values in a list set
- Example:
  - List all employees whose salaries are 17000, 2500, 8600, 1000

```
SELECT employee_id, last_name, salary
FROM employees
WHERE salary IN (17000, 2500, 8600, 1000);
```

```
SQL> select employee_id, last_name, salary
  2  from employees
  3  where salary in (17000, 2500, 8600, 1000);

EMPLOYEE_ID LAST_NAME                     SALARY
----------- ------------------------- ----------
        101 Kochhar                        17000
        102 De Haan                        17000
        144 Vargas                          2500
        176 Taylor                          8600
```

There is no employee with a salary of 1000. Therefore, it does not return a value.

SELECT…FROM…WHERE…LIKE (1)

- To perform wild card searchers of valid search string values
  - _ (underscore symbol) denotes one character
  - % denotes zero or many character
- Example:
  - List all employee whose first name starts with a J

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

```
SQL> select employee_id, first_name
  2  from employees
  3  where first_name like 'J%';

EMPLOYEE_ID FIRST_NAME
----------- --------------------
        176 Jonathon
        200 Jennifer
```

SELECT…FROM…WHERE…LIKE (2)

- Example:
- List all employees whose first name's third letter is an E

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

(2 underscore symbols)

```
SQL> select employee_id, first_name
  2  from employees
  3  where first_name like '__e%';

EMPLOYEE_ID FIRST_NAME
----------- --------------------
        205 Shelley
        103 Alexander
        100 Steven
        101 Neena
        141 Trenna
        149 Eleni

6 rows selected.
```

SELECT…FROM…WHERE…IS NULL

- To search for columns with **null value**
- Example:
  - List all employees who has not receive any commission.

```
SELECT employee_id,
first_name, commission_pct
FROM employees
WHERE commission_pct
IS NULL;
```

```
SQL> select employee_id, first_name, commission_pct
  2  from employees
  3  where commission_pct IS NULL;

EMPLOYEE_ID FIRST_NAME           COMMISSION_PCT
----------- -------------------- --------------
        200 Jennifer
        201 Michael
        202 Pat
        205 Shelley
        206 William
        100 Steven
        101 Neena
        102 Lex
        103 Alexander
        104 Bruce
        107 Diana

EMPLOYEE_ID FIRST_NAME           COMMISSION_PCT
----------- -------------------- --------------
        124 Kevin
        141 Trenna
        142 Curtis
        143 Randall
        144 Peter

16 rows selected.
```

SELECT…FROM…WHERE…IS NOT NULL

- To search for columns with value is not NULL
- Example:
  - List all employees who have received any commissions

```
SELECT employee_id, first_name, commission_pct
FROM employees
WHERE commission_pct IS NOT NULL;
```

```
SQL> select employee_id, first_name, commission_pct
  2  from employees
  3  where commission_pct IS NOT NULL;

EMPLOYEE_ID FIRST_NAME           COMMISSION_PCT
----------- -------------------- --------------
        149 Eleni                            .2
        174 Ellen                            .3
        176 Jonathon                         .2
        178 Kimberely                       .15
```

SELECT…FROM…WHERE…AND

- AND requires both component conditions to be true

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >= 10000        CONDITION 1
AND job_id LIKE '%MAN%';     CONDITION 2
```

```
SQL> select employee_id, last_name, job_id, salary
  2  from employees
  3  where salary >= 10000
  4  and job_id like '%MAN%';

EMPLOYEE_ID LAST_NAME                 JOB_ID         SALARY
----------- ------------------------- ---------- ----------
        201 Hartstein                 MK_MAN          13000
        149 Zlotkey                   SA_MAN          10500
```

SELECT…FROM…WHERE…OR

- OR requires either component conditions to be true

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >= 10000        CONDITION 1
OR job_id LIKE '%MAN%';      CONDITION 2
```

```
SQL> select employee_id, last_name, job_id, salary
  2  from employees
  3  where salary >= 10000
  4  or job_id like '%MAN%';

EMPLOYEE_ID LAST_NAME                 JOB_ID         SALARY
----------- ------------------------- ---------- ----------
        201 Hartstein                 MK_MAN          13000
        205 Higgins                   AC_MGR          12000
        100 King                      AD_PRES         24000
        101 Kochhar                   AD_VP           17000
        102 De Haan                   AD_VP           17000
        124 Mourgos                   ST_MAN           5800
        149 Zlotkey                   SA_MAN          10500
        174 Abel                      SA_REP          11000

8 rows selected.
```

NOT operator

- Example:
  - Display the last name and job ID of employees whose job ID is not IT_PROG, ST_CLERK, or SA_REP

```
SELECT last_name, job_id
FROM employees
WHERE job_id
NOT IN ('IT_PROG', 'ST_CLERK',
'SA_REP');
```

```
SQL> select last_name, job_id
  2  from employees
  3  where job_id
  4  not in ('IT_PROG', 'ST_CLERK', 'SA_REP');

LAST_NAME                 JOB_ID
------------------------- ----------
De Haan                   AD_VP
Fay                       MK_REP
Gietz                     AC_ACCOUNT
Hartstein                 MK_MAN
Higgins                   AC_MGR
King                      AD_PRES
Kochhar                   AD_VP
Mourgos                   ST_MAN
Whalen                    AD_ASST
Zlotkey                   SA_MAN

10 rows selected.
```

Rules of precedence

- You can use parenthesis to override rules of precedence

Operator	Meaning
1	Arithmetic operators
2	Concatenation operators
3	Comparison operators
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Not equal to
7	NOT logical condition
8	AND logical condition
9	OR logical condition

Rules of Precedence

1

```
SELECT last_name, job_id,salary
FROM employees
WHERE job_id = 'SA_REP'
OR job_id = 'AD_PRES'
AND salary > 15000;
```

Check this condition first

```
SQL> select last_name, job_id, salary
  2  from employees
  3  where job_id = 'SA_REP'
  4  OR job_id = 'AD_PRES'
  5  AND salary > 15000;

LAST_NAME                 JOB_ID          SALARY
------------------------- ---------- ----------
King                      AD_PRES         24000
Abel                      SA_REP          11000
Taylor                    SA_REP           8600
Grant                     SA_REP           7000
```

2

```
SELECT last_name, job_id,salary
FROM employees
WHERE (job_id = 'SA_REP'
OR job_id = 'AD_PRES')
AND salary > 15000;
```

Check the one with parenthesis first

```
SQL> select last_name, job_id, salary
  2  from employees
  3  where (job_id = 'SA_REP'
  4  or job_id = 'AD_PRES')
  5  and salary > 15000;

LAST_NAME                 JOB_ID          SALARY
------------------------- ---------- ----------
King                      AD_PRES         24000
```

DISTINCT (1)

- The default display of queries is all rows, including duplicate rows
- Use the DISTINCT keyword immediately after the SELECT keyword to eliminate duplicate rows

```
SELECT DISTINCT department_id
FROM employees;
```

```
SQL> select department_id
  2  from employees;

DEPARTMENT_ID
-------------
           10
           20
           20
          110
          110
           90
           90
           90
           60
           60
           60

DEPARTMENT_ID
-------------
           50
           50
           50
           50
           50
           80
           80
           80

20 rows selected.
```

```
SQL> select distinct department_id
  2  from employees;

DEPARTMENT_ID
-------------

           20
           90
          110
           50
           80
           10
           60

8 rows selected.
```

DISTINCT (2)

- Multiply columns can be specified after the DISTINCT qualifier
- This affects all the selected columns, and the result is every distinct combination of the columns

```
SELECT DISTINCT department_id,
job_id FROM employees;
```

```
SQL> select distinct department_id, job_id
  2  from employees;

DEPARTMENT_ID JOB_ID
------------- ----------
          110 AC_ACCOUNT
           90 AD_VP
           50 ST_CLERK
           80 SA_REP
          110 AC_MGR
           50 ST_MAN
           80 SA_MAN
           20 MK_MAN
           90 AD_PRES
           60 IT_PROG
              SA_REP

DEPARTMENT_ID JOB_ID
------------- ----------
           10 AD_ASST
           20 MK_REP

13 rows selected.
```

DESCRIBE

- Use the DESCRIBE command to **display** the **structure** of a table

```
SQL> describe employees;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
 EMPLOYEE_ID                               NOT NULL NUMBER(6)
 FIRST_NAME                                         VARCHAR2(20)
 LAST_NAME                                 NOT NULL VARCHAR2(25)
 EMAIL                                     NOT NULL VARCHAR2(25)
 PHONE_NUMBER                                       VARCHAR2(20)
 HIRE_DATE                                 NOT NULL DATE
 JOB_ID                                    NOT NULL VARCHAR2(10)
 SALARY                                             NUMBER(8,2)
 COMMISSION_PCT                                     NUMBER(2,2)
 MANAGER_ID                                         NUMBER(6)
 DEPARTMENT_ID                                      NUMBER(4)
```

SUMMARY

- **INSERT INTO** : To add new values (data) into a table
- **UPDATE … SET** : To update existing value of a column with a new value
- **SELECT … FROM** : To retrieve records in table(s)
- **SELECT … FROM …. WHERE** : To retrieve record that match certain condition

Simple Handwritten Exercise

- Given the following relation schemas, construct the SQL statement for following tasks:

CUSTOMER (**customer_id**, store_id, first_name, last_name, email, address, active)

RENTAL (**rental_id**, rental_date, inventory_id, customer_id, return_date, staff_id)

INVENTORY (**inventory_id**, film_id, store_id)

STORE (**store_id**, manager_staff_id, location)

STAFF (**staff_id**, first_name, last_name, address, email, store_id, salary)

PAYMENT (**payment_id**, customer_id, staff_id, rental_id, amount, payment_date)

FILM (**film_id**, title, description, rental_duration, rental_rate)

1) List all films title, rental duration and rental rate.
2) List all customers' details who registered to store with ID of ST_002.
3) Insert a new film in database which has the film ID 'F_0099', title 'My Neighbour Totoro', description 'Is a 1998 Japanese animated film by Hayao Miyazaki', rental duration 14 days, and rental rate of RM8.