Content

- Introduction to SQL
- Categories of SQL Commands: DDL, DML, DCL, DTL/TCL
- DDL (CREATE/ALTER/DROP/TRUNCATE)
- DML (INSERT/UPDATE/DELETE)
- MySQL Data Types
- Database Constraints (Primary Key, Unique, Not Null, Foreign Key, Default, Check*)
- Aggregate Functions, Grouping Things Together (Group By, Having)
- LIKE Operator, DISTINCT, Sorting (Order by clause)
- BETWEEN AND Operators, Comparing Nulls (IS NULL/IS Not NULL), IN/NOT IN

What is SQL ?

- *Structured Query Language*.
- Access mechanism to a relational database and therefore at the heart of any contemporary RDBMS.

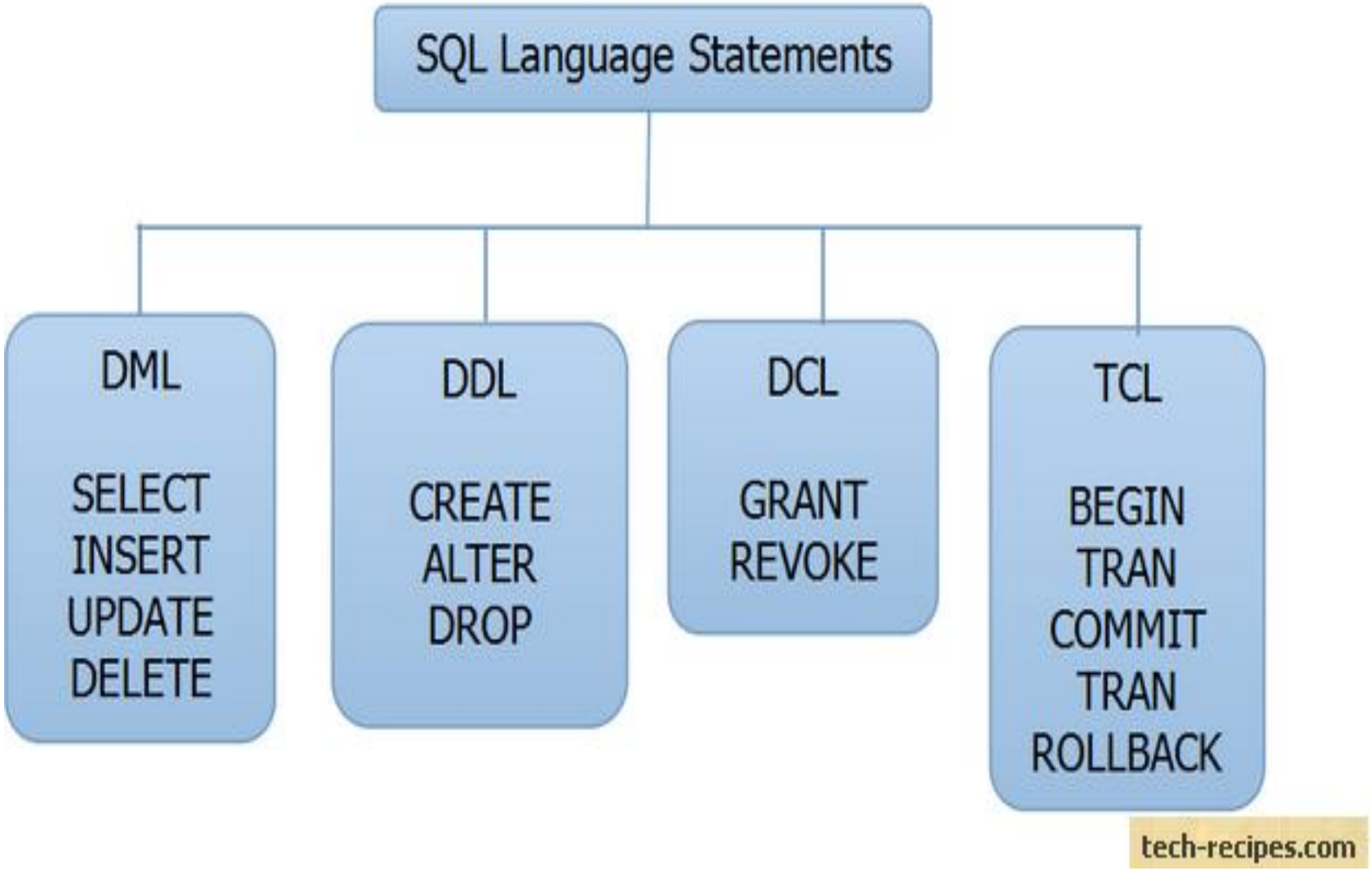
Components of SQL:

- Data Definition Component
  a.k.a. Data Definition Language (DDL)
- Data Manipulation Component
  a.k.a. Data Manipulation Language (DML)
- Transaction Control Component
- Others

  View Definition, Embedded DML, Integrity

SQL in action:

- Single table queries
- Multiple table queries
- Aliases
- Subqueries
- Aggregate functions
- Advanced SQL features


SQL Language Statements
DML
SELECT
INSERT
UPDATE
DELETE
DDL
CREATE
ALTER
DROP
DCL
GRANT
REVOKE
TCL
BEGIN
TRAN
COMMIT
TRAN
ROLLBACK
tech-recipes.com

Data Definition Language (DDL) Statements

- The CREATE, ALTER, and DROP commands require exclusive access to the specified object.
  - ALTER TABLE statement fails if another user has an open transaction on the specified table.

- The GRANT, REVOKE, ANALYZE, AUDIT, and COMMENT commands do not require exclusive access to the specified object.
  - You can analyze a table while other users are updating the table.

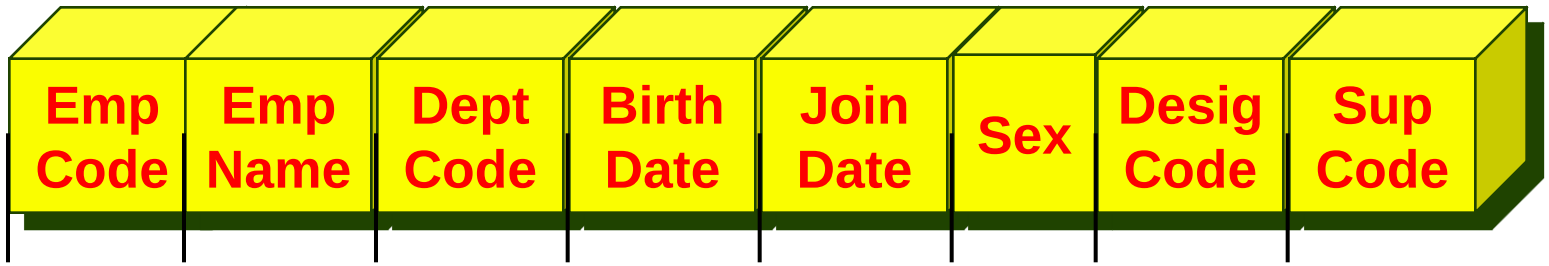
Data Manipulation Language (DML) Statements

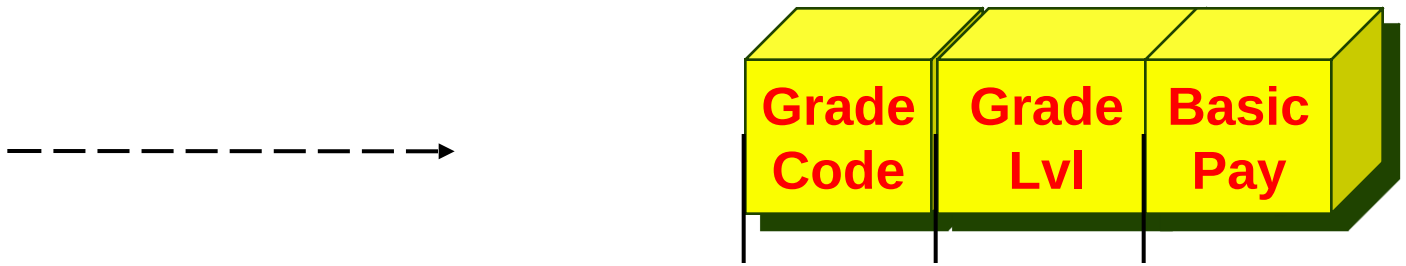
- Data manipulation language (DML) statements access and manipulate data in existing schema objects.
- These statements do not implicitly commit the current transaction.
- The SELECT statement is a limited form of DML statement in that it can only access data in the database

Transaction control statements

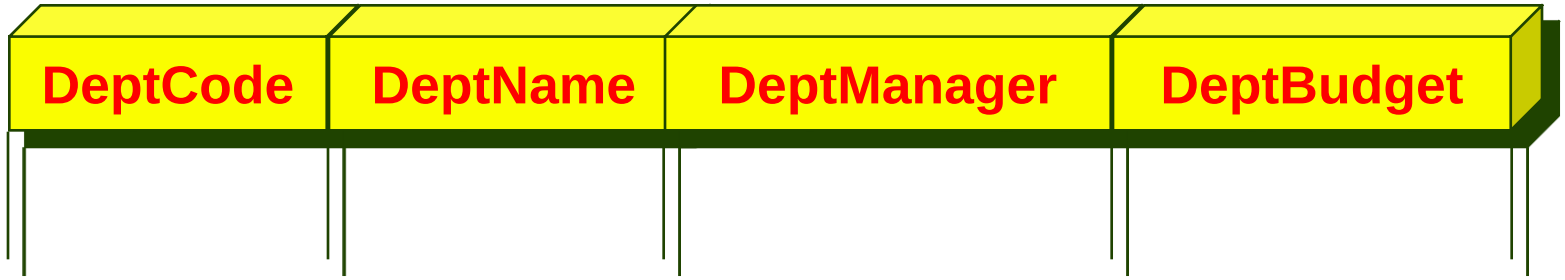
- Transaction control statements manage changes made by DML statements.

Emp Table

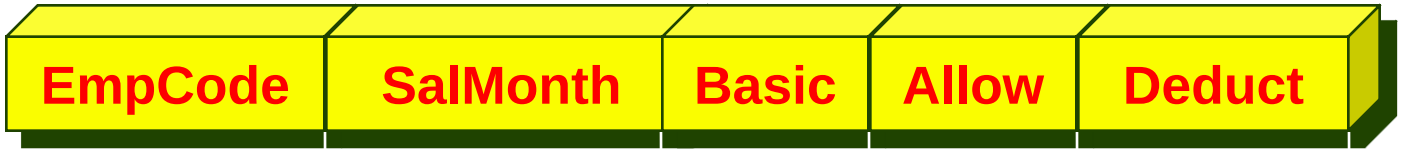

Emp Code
Emp Name
Dept Code
Birth Date
Join Date
Sex
Desig Code
Sup Code


Grade Code
Grade Lvl
Basic Pay

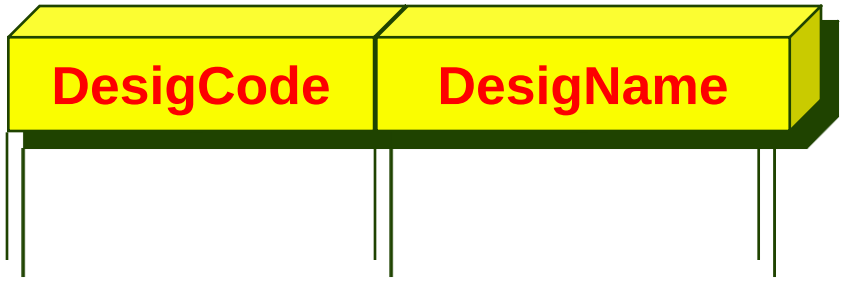
Dept Table



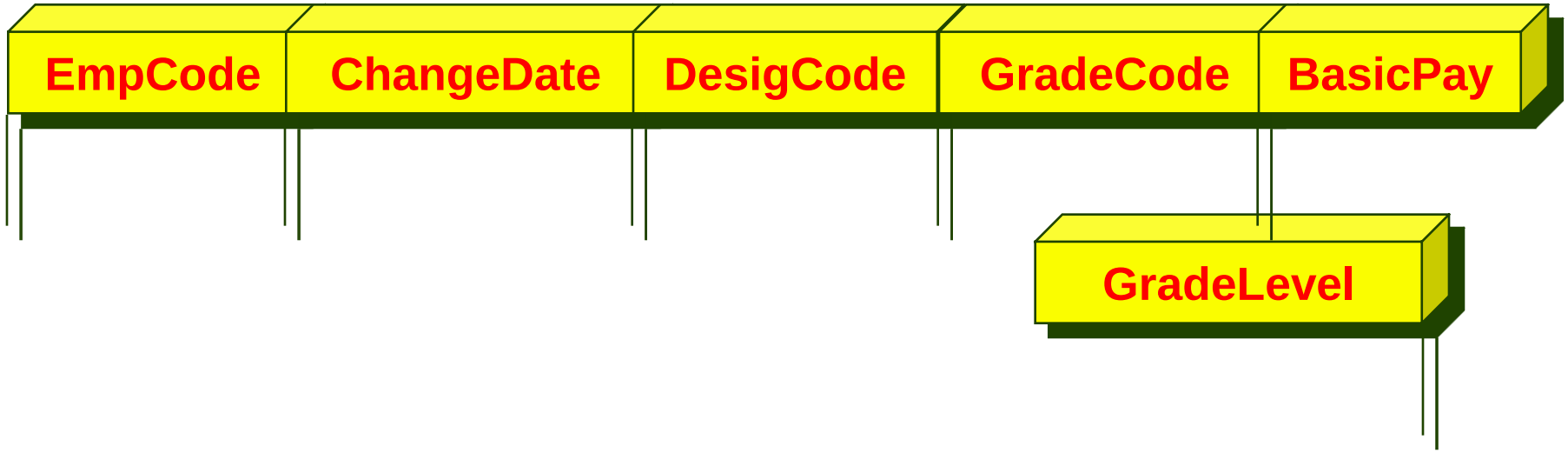
Salary Table



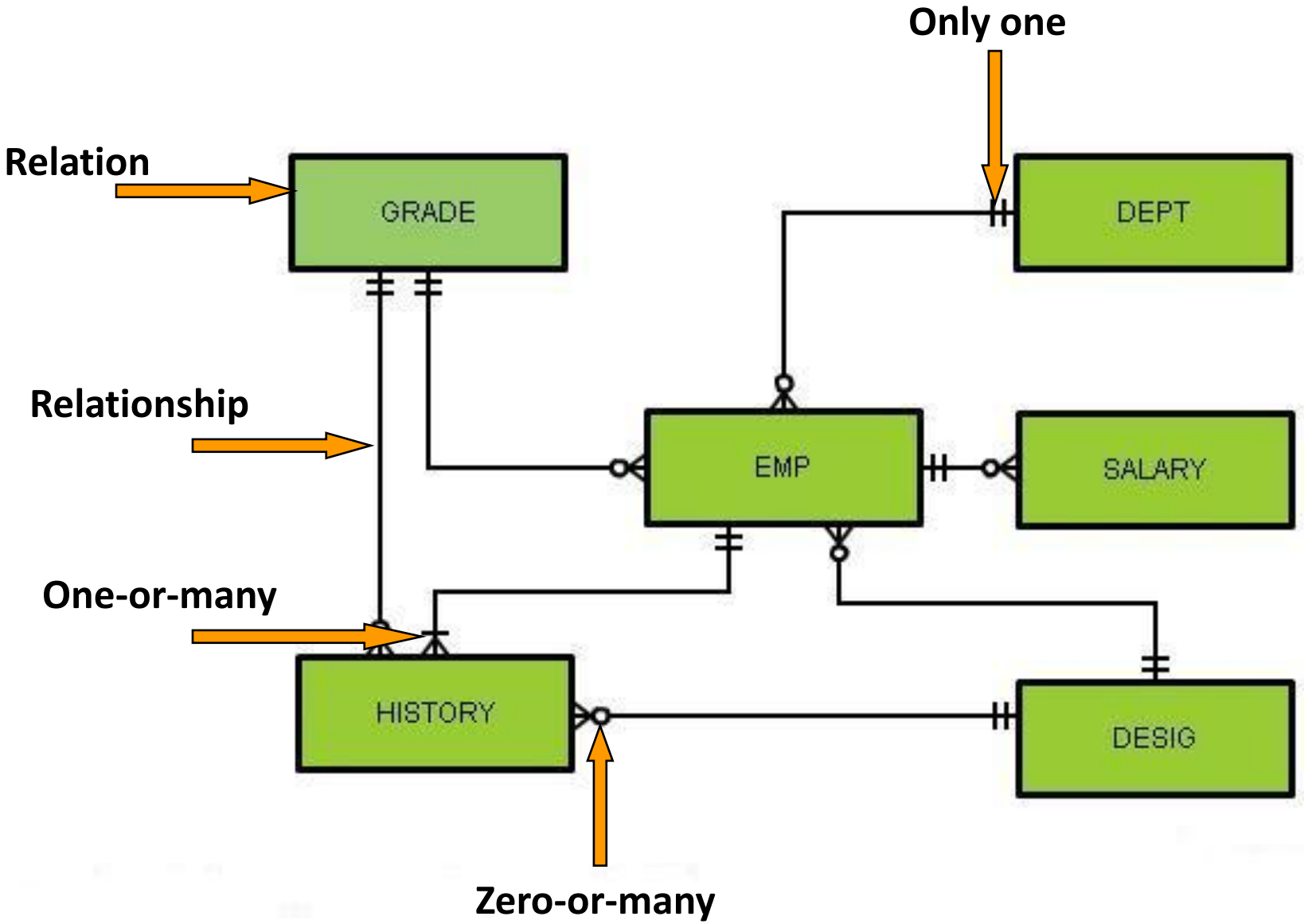
Desig Table



History Table



Schema – ER Diagram



The Whole Table

1 **List the Department table**

```
SELECT   *
FROM   Dept;
```

```
DEPT DEPTNAME    DEPTMA DEPTBUDGET
---- ---------- ------ ----------
ACCT Accounts     7839         19
PRCH Purchase     7902         25
SALE Sales        7698         39
STOR Stores       7521         33
FACL Facilities   7233         42
PERS Personnel    7233         12

6 rows selected.
```

Another Way - Better!

2 **List the Department table**

```
SELECT DeptCode, DeptName,
       DeptManager, DeptBudget

FROM   Dept;
```

DEPT	DEPTNAME	DEPTMA	DEPTBUDGET
ACCT	Accounts	7839	19
PRCH	Purchase	7902	25
SALE	Sales	7698	39
STOR	Stores	7521	33
FACL	Facilities	7233	42
PERS	Personnel	7233	12

6 rows selected.

Only Some Columns

3 **List all department managers with the names of their departments**

```
SELECT DeptManager, DeptName
FROM   Dept;
```

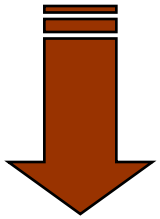
```
DEPTMA DEPTNAME
------ ------------------------
7839   Accounts
7902   Purchase
7698   Sales
7521   Stores
7233   Facilities
7233   Personnel

6 rows selected.
```

SQL query structure:

SELECT $A_1, A_2, \ldots, A_n$

FROM $r_1, r_2, \ldots, r_m$

WHERE $P;$



$$\Pi_{A_1, A_2, \ldots, A_n} (\sigma_P (r_1 \times r2 \times \ldots \times r_m))$$

DISTINCT

4 List all department managers

```
SELECT    DeptManager
FROM        Dept;
```

duplicates are NOT eliminated !

```
DEPT
----
Khan
Khan
Roy
Patil
Khan
```

DISTINCT

5

List all department managers

```
SELECT DISTINCT DeptName
FROM Dept;
```

```
DEPT
----
Khan
Roy
Patil
```

WHERE Predicate (=)

6 **List all employees of the Accounts department**

```
SELECT      EmpName, DeptCode
FROM        Emp
WHERE       DeptCode = 'ACCT';
```

```
EMPNAME     DEPT
----------  ----
Reddy       ACCT
Menon       ACCT
Kaul        ACCT
```

WHERE Predicate (=)

7 **List all officers**

```
SELECT   EmpName, GradeCode
FROM     Emp
WHERE    GradeCode = 'GC6';
```

EMPNAME	GRADECODE
Naik	4
Reddy	1
Murthy	4
Wilson	4
Jain	4
Menon	4
Khan	6
Kumaran	4
Kamal	4

9 rows selected.

SELECT from more than a table

8 **List the employees with their department name**

```
SELECT    EmpName, DeptName
FROM      Emp, Dept
WHERE     Emp.DeptCode = Dept.DeptCode;
```

SELECT from more than a table

8 **List the employees with their department code and department name**

```
SELECT    EmpName, DeptName, Dept.DeptCode
FROM      Emp, Dept
WHERE     Emp.DeptCode = Dept.DeptCode;
```

```
EMPNAME    DEPTOCDE   DEPTNAME
---------- ---------  ------------
Shah       PRCH       Purchase
Naik       PRCH       Purchase
Reddy      ACCT       Accounts
Jain       PRCH       Purchase
Menon      ACCT       Accounts
 .          .          .
 .          .          .
 .          .          .

17 rows selected.
```

Date Comparison

9 **List all young employees**

```
select empname,birthdate
from emp
where birthdate > '1980-01-01‘;
```

Set Membership

10 List employees of admin departments

```
SELECT    EmpName, DeptCode
FROM      Emp
WHERE     DeptCode IN  ('ACCT' , 'PRCH' , 'PERS' );
```

EMPNAME	DEPT
Shah	PRCH
Naik	PRCH
Reddy	ACCT
Jain	PRCH
Menon	ACCT
Khan	PRCH
Patil	PRCH
Kaul	ACCT
Uma	PERS

9 rows selected.

Set Membership

11 List employees of non-admin departments

```
SELECT  EmpName, DeptCode
FROM    Emp
WHERE   DeptCode NOT IN ('ACCT' , 'PRCH' , 'PERS' );
```

NOT IN

Between Construct

12 **List staff between certain age**

```
select empname,birthdate
from emp
where birthdate between '1982-12-31' and '1992-12-31';
```

EMPNAME	BIRTHDATE
Rai	08/10/1988
Tiwari	08/19/1989

Like Construct

13 **List all employees with UMA in their names**

```
SELECT    EmpName
FROM      Emp
WHERE     Upper(EmpName) LIKE '%UMA%';
```

LIKE

```
EMPNAME
----------
Kumaran
Uma
```

Null Values

14 **List the employees who have not been assigned to any supervisor**

```
SELECT   EmpName, SupCode
FROM     Emp
WHERE    SupCode IS NULL;
```

IS NULL

```
EMPNAME    SUPCOD
---------- ------
 Reddy
```

Compound Predicate - Interval

15 **List the female employees who have just completed 5 years**

```
SELECT   EmpName, Sex, JoinDate
FROM     Emp
WHERE    Sex = 'F' AND    AND
         JoinDate BETWEEN (CURRENT_DATE - 5*365)
                      AND (CURRENT_DATE - 6*365);
```

```
EMPNAME     S JOINDATE
---------- - ---------
Uma         F 22-OCT-91
```

Predicate using OR

16 **List FEMALE employees who are either 50 years or more or have more than 20 years experience**

```
SELECT   EmpName
FROM     Emp
WHERE    BirthDate < (CURRENT_DATE - 50*365)
         OR JoinDate < (CURRENT_DATE - 20*365) and SEX= 'F';
```

```
EMPNAME
----------
Reddy
Kumaran
Kamal
```

Parentheses in Predicate

17 List salesmen who are either 50 years or more or have more than 20 years' experience

```
SELECT  EmpName
FROM    Emp
WHERE   DesigCode = 'SLMN'
AND     (BirthDate < (CURRENT_DATE - 50*365)
                          OR
         JoinDate < (CURRENT_DATE - 20*365));
```

```
EMPNAME
----------
Kumaran
Kamal
```

Expressions in SELECT List

18 **List 1% of take-home pay of all employees**

```
SELECT    EmpCode, (Basic + Allow - Deduct) * 0.01
FROM      Salary
WHERE     SalMonth = '02/01/2012';
```

```
EMPCOD (BASIC+ALLOW-DEDUCT)*0.01
------ -------------------------
7129                          440
7233                          440
7345                          143
7369                           66
.                                .
.                                .
.                                .
7844                          198
7876                           44
7900                           55
7902                          330
7934                           77

17 rows selected.
```

Aliasing

19 **List the present age of all the employees**

```
select empname,timestampdiff(year,birthdate,curdate()) as age
 from emp;
```

Defining a Column Alias

- A column alias:
  - Renames a column heading
  - Is useful with calculations
  - Immediately follows the column name - there can also be the optional AS keyword between the column name and alias

```
EMPNAME           AGE
---------- ---------
Shah               20
Naik               36
Reddy              54
Murthy             34
Roy                24
.                    .
.                    .
.                    .
Shroff             19
Kaul               21
Kumaran            57
Kamal              46
Uma                22

17 rows selected.
```

ALL construct

20 **List the employees with the highest salary**

```
SELECT   EmpName, basicpay
FROM     Emp
WHERE    BasicPay >= ALL( SELECT   BasicPay
                          FROM     Emp);
```

```
EMPNAME    BasicPay
---------- ---------
 Reddy     40000
```

- The ALL,ANY comparison condition is used to compare a value to a list or subquery.
- It must be preceded by =, !=, >, <, <=, >= and followed by a list or subquery.
- When the ALL condition is followed by a list, the optimizer expands the initial condition to all elements of the list and strings them together with AND operators
- When the ANY condition is followed by a list, the optimizer expands the initial condition to all elements of the list and strings them together with OR operators

ALL

- Page > ALL(4,2,7) – Page is greater than items in the list (4,2,7) – anything larger than 7 qualifies.
- Page < ALL(4,2,7) – Page less than lowest items in the list (4,2,7) – anything less than 2 qualifies.
- Page != ALL(4,2,7) – Page not equal to any items in the list (4,2,7) – any number qualifies except 4,2 and 7.

ANY

- Page > ANY(4,2,7) – Page is greater than any single items in the list (4,2,7)
    - 3 qualifies, because it is greater than 2
- Page < ANY(4,2,7) – Page is less than any single items in the list (4,2,7)
    - 6 qualifies as it is less than 7.
- Page != ANY(4,2,7) – Page not equal to any single item in the list (4,2,7)

ANY construct

21 **List all the employees not having the highest salary**

```
SELECT   EmpName
FROM     Emp
WHERE    BasicPay < ANY (SELECT   BasicPay
                         FROM     Emp);
```

UNION and UNION ALL

22 **List the employees working for ‘accounts’ or ‘purchase’ departments**

```
SELECT   EmpName
FROM      Emp
WHERE    DeptCode = ‘ACCT’
            UNION
SELECT   EmpName
FROM      Emp
WHERE    DeptCode = ‘PRCH’;
```

To show duplicates, use UNION ALL

EMPNAME	DEPT
Shah	PRCH
Naik	PRCH
Reddy	ACCT
Jain	PRCH
Menon	ACCT
Khan	PRCH
Patil	PRCH
Kaul	ACCT

8 rows selected.

Restriction on UNION, INTERSECT and MINUS

- Queries that use UNION, INTERSECT and MINUS in where clause must have the same number and type(data type) of columns in their select list
- Only the column names from the first select statement can be used in **order by** clause.
- **IN** construction does not have this limitation

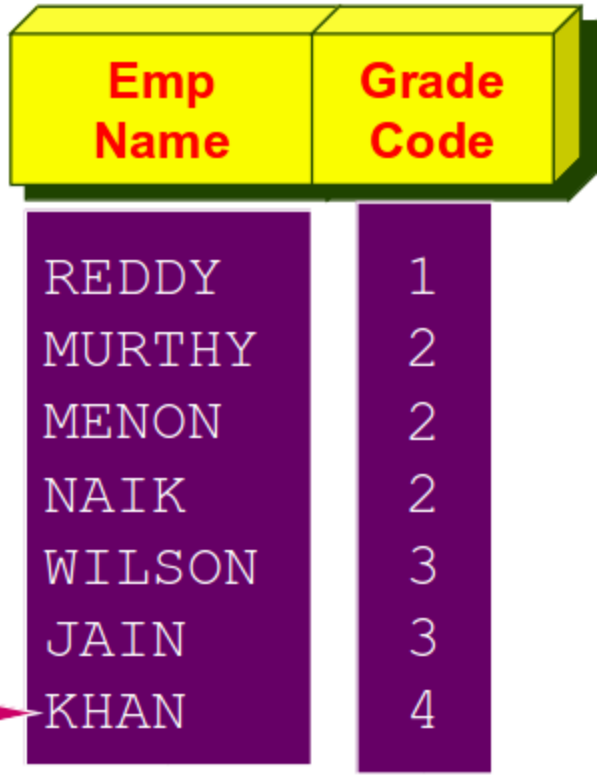
ORDER BY

Emp Name	Grade Code
WILSON	3
JAIN	3
MURTHY	2
MENON	2
KHAN	4
REDDY	1
NAIK	2

No particular order

ORDER BY

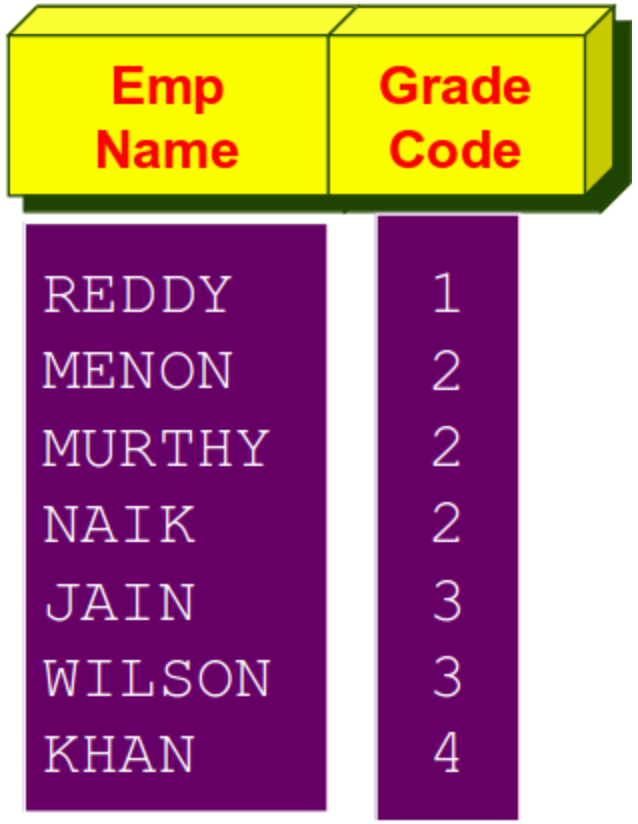
```
SELECT      .....
FROM        .....
.....
.....
ORDER BY  GradeCode;
```



No ordering of EmpName

ORDER BY

```
SELECT      .......
FROM        .......
.......
......
ORDER BY    GradeCode, EmpName;
```



Order by

25 **List all employees ordered by age**

```
SELECT      EmpName,
      TRUNC((CURRENT_DATE - BirthDate) / 365) as AGE
FROM        Emp
ORDER BY    BirthDate;
```

```
select empname,timestampdiff(year,birthdate,curdate()) as age
from emp
order by birthdate;
```

```
EMPNAME              AGE
---------- ---------
Kumaran               57
Reddy                 54
Kamal                 46
Menon                 38
.                      .
.                      .
.                      .
Uma                   22
Patil                 21
Kaul                  21
Shah                  20
Shroff                19

17 rows selected.
```

Sorting Ascending/Descending

26 **List middle level staff according to seniority**

```
SELECT      EmpName, GradeCode, GradeLevel
FROM        Emp
WHERE       GradeCode BETWEEN 10 AND 25
ORDER BY    GradeCode, GradeLevel DESC;
```

EMPNAME	GRADECODE	GRADELEVEL
Gupta	12	3
Roy	12	2
Singh	12	1
Uma	15	2
Patil	20	4
Shroff	20	3
Shah	20	2
Kaul	20	1

8 rows selected.

Aggregate Functions

28 **Count employees reporting to Singh**

```
SELECT   COUNT(*)
FROM     Emp
WHERE    SupCode = '7839';
```

```
 COUNT(*)
---------
        1
```

Aggregate Functions

29 **Count employees reporting to Singh**

```
SELECT   COUNT(*)
FROM     Emp
WHERE    SupCode = 'empcode of Singh';
```

Aggregate Functions

30 **Count employees reporting to Singh**

```
SELECT   COUNT(*)
FROM     Emp
WHERE    SupCode = (SELECT EmpCode
                    FROM EMP
                    WHERE EmpName = 'Singh');
```

```
 COUNT(*)
---------
         1
```

Group By - Aggregate Functions

31 **List the number of staff reporting to each supervisor**

```
SELECT      SupCode, COUNT(*)
FROM        Emp
GROUP BY    SupCode
ORDER BY    SupCode;
```

```
 SUPCOD  COUNT(*)
------ ----------
7566             1
7698             5
7782             1
7788             1
7839             6
7844             1
7902             1
                 1

8 rows selected.
```

List the total take-home pay during 96-97 for all employees

- select empcode, sum(basic+allow-deduct) as pay
 from salary where salmonth between '2011-01-12' and '2012-01-12‘
 group by empcode
order by empcode;

```
EMPCOD        PAY
------ ----------
7129       132000
7233       132000
7345        42900
7369        19800
7499        56100
.
.
.
7876        13200
7900        16500
7902        99000
7934        15400

17 rows selected.
```

Group By - Max & Min

33 **List the maximum & minimum salaries in grades**

```
SELECT      GradeCode, MAX(Basic), MIN(Basic)
FROM        Grade
GROUP BY    GradeCode
ORDER BY    GradeCode;
```

GRADECODE	MAX(BASIC)	MIN(BASIC)
1	25000	25000
4	21000	15000
6	13000	11000
12	9000	8000
15	7000	6000
20	3500	2000

6 rows selected.

Having

Having clause is similar to where clause, except it's logic is only related to result of group function, as opposed to columns or individual rows, which can still be selected by a where clause.

Having

34

List the number of staff reporting to each supervisor having more than 3 people working under them

```
SELECT      SupCode, COUNT(*)
FROM        Emp
GROUP BY    SupCode
HAVING      COUNT(*) > 3
ORDER BY    SupCode;
```

```
SUPCOD   COUNT(*)
------ ----------
7698              5
7839              6
```

Where - Group By - Having

35

List the total take-home pay during 2011 -2012 for all employees getting a total take-home-pay < Rs. 20000

```
SELECT      EmpCode, SUM( Basic + Allow - Deduct)
as PAY
FROM        Salary
WHERE       SalMonth BETWEEN '12/01/2011'
            AND '12/01/2012'
GROUP BY    EmpCode
HAVING      SUM( Basic + Allow - Deduct) < 40000
ORDER BY    EmpCode;
```

Where - Group By - Having

EMPCODE	PAY
6569	38180
7192	37760
7369	37490
7521	37440
7566	38380
7654	38720
7782	38080
7788	36530
7802	38280
7876	37760
7900	37000
7902	36210
7934	36890
7939	36260

Where - Group By - Having

37 **List the maximum and minimum basic salary in each grade for grades with start < Rs. 4000**

```
SELECT GradeCode, MAX(Basic), MIN(Basic)
FROM Grade
GROUP BY GradeCode
HAVING MIN(Basic) < 4000
ORDER BY GradeCode;
```

```
GRADECODE MAX(BASIC) MIN(BASIC)
--------- ---------- ----------
       20       3500       2000
```

Order of execution

1. Choose rows based on where clause
2. Group rows together based on group by clause
3. Calculate result of group functions for each group
4. Choose and eliminate groups based on having clause
5. Order the group based on result of group function in the Order by clause.
7. The order by clause must use either a group function or a column specified in the group by clause.

Order of execution has impact on performance of the query.
a.If more record can be eliminated by where clause faster the query will perform, because less number of rows will be Processed in group by clause.

INNER JOIN

40 List employees along with the names of their supervisors

```
SELECT  E.EmpCode, E.EmpName,
        S.EmpCode, S.EmpName

FROM    Emp E INNER JOIN Emp S
ON      E.SupCode = S.EmpCode;
```

EMPCOD	EMPNAME	EMPCOD	EMPNAME
7369	Shah	7902	Naik
7902	Naik	7839	Reddy
7698	Murthy	7839	Reddy
7499	Roy	7698	Murthy
7521	Wilson	7698	Murthy
.			
.			
.			
7900	Shroff	7698	Murthy
7934	Kaul	7782	Menon
7233	Kumaran	7839	Reddy
7129	Kamal	7839	Reddy
7345	Uma	7844	Singh

16 rows selected.

INNER JOIN :

38 **List employees along with their basic salary**

```
SELECT   EmpCode, EmpName, Salary.Basic
FROM     Emp INNER JOIN Salary
USING    (EmpCode);
```

Inner joins are default, it return rows of two table have in common.
On/ Using- To specify join criteria

NATURAL JOIN :

39 **List employees along with their basic salary**

```
SELECT   EmpCode, EmpName, Basic
FROM     Emp NATURAL JOIN Salary ;
```

Natural – Join should be performed based on all columns that have the same names, in the two tables being joined

EMPCOD	EMPNAME	BASIC
7369	Shah	3000
7902	Naik	15000
7839	Reddy	25000
7698	Murthy	17000
7499	Roy	8500
.		
.		
.		
7934	Kaul	3500
7233	Kumaran	20000
7129	Kamal	20000
7345	Uma	6500

17 rows selected.

INNER JOIN (Self Join):

41 **List employees along with the names of their supervisors**

```
SELECT  E.EmpCode, E.EmpName,
        S.EmpCode, S.EmpName

FROM    Emp E INNER JOIN Emp S
ON      E.SupCode = S.EmpCode;
```

INNER JOIN : Where clause

42 List employees along with the names of their department for which they are working

```
SELECT   E.EmpCode, E.EmpName, D.DeptName

FROM     Emp E, Dept D
WHERE    E.DeptCode = D.DeptCode;
```

```
EMPCOD EMPNAME    DEPTCODE
------ ---------- --------
7369   Shah       PRCH
7902   Naik       PRCH
7698   Murthy     SALE
7499   Roy        SALE
7521   Wilson     STOR
.
.
.
7900   Shroff     SALE
7129   Jain       PRCH

20 rows selected.
```

RIGHT Outer **JOIN** :

43

List employees along with the names of their department for which they are working. The list should have all the departments listed.

```
SELECT  E.EmpName, D. DeptCode

FROM    Emp E RIGHT OUTER JOIN Dept D
ON      E.DeptCode = D.DeptCode;
```

RIGHT Outer **JOIN** :

43

List employees along with the names of their department for which they are working. The list should have all the departments listed.

```
SELECT  E.EmpName, D. DeptCode

FROM    Emp E RIGHT OUTER JOIN Dept D
USING   (DeptCode);
```

```
EMPCOD EMPNAME    DEPTCODE
------ ---------- --------
7369   Shah       PRCH
7902   Naik       PRCH
7698   Murthy     SALE
7499   Roy        SALE
7521   Wilson     STOR
.
.
.
7900   Shroff     SALE
7129   Jain       PRCH
----     ----       PERS
----   ----       RCMT
----     ----       FACL

23 rows selected.
```

Full Outer **JOIN** :

44

```
SELECT  E.EmpName, E.DeptCode, D.DeptCode

FROM      Emp E FULL OUTER JOIN Dept D
ON        E.DeptCode  = D.DeptCode;
```

46

List the number of officers reporting to each supervisor having more than 3 people working under them.

```
SELECT SupCode, COUNT(*)
FROM   Emp
WHERE GradeCode < 10
AND    SupCode IN
(SELECT SupCode
 FROM   Emp
 GROUP BY SupCode
 HAVING COUNT(*) > 3 )
GROUP BY Supcode;
```

```
SUPCOD  COUNT(*)
------ ---------
7698             1
7839             6
```

EXISTS

47 List employees who did not get any promotion since 1990.

```
SELECT     EmpCode, EmpName, DeptCode
FROM          Emp
WHERE     NOT EXISTS
       (promotion records for him since 1990);
```

EXISTS

48 List employees who get any promotion since 1990.

```
SELECT    EmpCode, EmpName, DeptCode
FROM          Emp E
WHERE     NOT EXISTS
       (SELECT  *
         FROM     History
         WHERE  E.EmpCode = EmpCode
         AND       changeDate>= date '1990-01-01');
```

Nested Queries

49 **List employees who were promoted to officer grade in 1995.**

```
SELECT  EmpCode, EmpName, DeptCode,
        GradeCode, GradeLevel
FROM    Emp
WHERE   EmpCode IN  ( SELECT   EmpCode
  FROM   History
  WHERE  GradeCode >= 10
  AND    changeDate BETWEEN '01-Jan-95'
                        AND '31-Dec-95');
```

```
EMPCOD EMPNAME     DEPT GRADECODE GRADELEVEL
------ ---------- ---- --------- ----------
7369   Shah        PRCH         20          2
```

Nested Queries

50 **List employees who did not get any promotion since 1990.**

```
SELECT  EmpCode, EmpName, DeptCode
FROM      Emp
WHERE   NOT EXISTS ( SELECT     *
      FROM    History
      WHERE   EMP.EmpCode = EmpCode
      AND changeDate > '01-Jan-90');
```

EMPCOD	EMPNAME	DEPT
7788	Khan	PRCH
7876	Patil	PRCH
7233	Kumaran	FACL

Nested Queries

**List the second highest Salary.**

```
SELECT  MAX (basic+allow-deduct) as takehome
FROM      SALARY
WHERE   (basic+allow-deduct)
NOT IN (SELECT MAX (basic+allow-deduct)
 FROM SALARY);
```

Other DML commands

- INSERT
- UPDATE
- DELETE
- MERGE

INSERT one complete row

51 **Promote Shah as Salesman.**

```
INSERT INTO  History
VALUES  ('7369','01-Aug-96','SLMN','12',5000);
```

INSERT one partial row

52 **Employ Hussein as a temporary employee.**

```
INSERT INTO     Emp
(EmpCode, EmpName, Basic)
VALUES
('9123', 'Hussein', 250);
```

INSERT thru' Subquery

53 **Update the SALARY table for the month.**

```
INSERT INTO     Salary
SELECT  EmpCode, CURRENT_DATE,
      Basic, Basic*1.5, Basic*0.3
FROM      Emp;
```

Delete one row

54 **Delete employee record of Kaul.**

```
DELETE  FROM  Emp
WHERE   EmpCode = '7934';
```

Bulk Delete

55 **Delete all employee records of Filing Department.**

```
DELETE FROM     Emp
WHERE DeptCode = 'FLNG';
```

Delete entire Table contents

56 **Delete the entire contents of SALARY table.**

```
DELETE FROM Salary;
```

Update one Row

57 **Promote Gupta as Manager(Exports).**

```
UPDATE  Emp
SET    GradeCode = '4',
DesigCode = 'MNGR',
Basic = 15000
WHERE   EmpCode = '7654';
```

Bulk Update

58

Raise the budget by 25% for all the departments except Facilities department.

```
UPDATE          Dept
SET     DeptBudget = DeptBudget * 1.25
WHERE           DeptCode != 'FACL';
```

DDL commands

- CREATE
- ALTER
- DROP
- TRUNCATE

CREATE TABLE

59 **Create Department table**

```
CREATE TABLE Dept (
      DeptCode          varchar2 (4)
                                constraint dept_pk primary key,
      DeptName          varchar2 (25) NOT NULL,
      DeptManager       varchar2 (6),
      DeptBudget        number NOT NULL);
```

CREATE TABLE

60 **Create Employee table**

```
CREATE TABLE Emp (
      EmpCode       varchar2(6)
                              constraint emp_pk primary key,
      EmpName       varchar2(20) not null,
      DeptCode      varchar2 (4)
                        constraint emp_dept_rc
                              references dept(deptcode),
```

Cntd ....

CREATE TABLE

```
BirthDate        Date not null,
JoinDate             Date not null,
Sex              char(1) not null check (sex in ('M', 'F')),
DesigCode        char(4) not null constraint emp_desig_rc
                                  references desig(DesigCode) ,
SupCode              char(6) constraint emp_sup_rc
                                  references emp(EmpCode) ,
GradeCode        number(2),
Basic                number(5));
```

CREATE TABLE

61 **Create Grade table**

```
CREATE TABLE Grade (
GradeCode       number(2) not null,
GradeLevel      number    not null,
Basic                 number(5) not null,
                      constraint grade_pk
                      primary key (GradeCode,GradeLevel)) ;
```

Table-level constraint

CREATE TABLE - Copying table structure/data

61 **Create Grade table**

CREATE TABLE GradeNew Like Grade;

This will copy the structure and indices, but not the data:

CREATE TABLE GradeNew AS SELECT * FROM Grade;

This will copy the data and the structure, but not the indices

INSERT INTO GradeNew SELECT * FROM Grade;

To Make structure of new table similar to previous

Create Table – Auto increment

- The attribute to use when you want to assign a sequence of numbers automatically to a field
- A table in MySQL can only contain one AUTO_INCREMENT column

```
CREATE TABLE contacts
( contact_id INT(11) NOT NULL AUTO_INCREMENT,
 last_name VARCHAR(30) NOT NULL,
first_name VARCHAR(25),
birthday DATE,
CONSTRAINT contacts_pk PRIMARY KEY (contact_id) );
```

Delete a table from the database :

62 **Drop the entire SALARY table.**

```
DROP TABLE Salary;
```

ALTER TABLE

63 **Drop column ‘basic’ from Grade**

```
ALTER TABLE Grade
DROP COLUMN Basic;
```

ALTER TABLE

64 **Add column ‘GradeInc’ to Grade**

```
ALTER TABLE Grade
ADD COLUMN GradeIncent number;
```

ALTER TABLE

65 **Increase the size of the field SupCode in Emp table**

```
ALTER TABLE Emp
MODIFY COLUMN SupCode  char(10);
```

Updating Rows in a Table

- Specific row or rows are modified if you specify the WHERE clause.
- All rows in the table are modified if you omit the WHERE clause.

- Update emp set deptcode='SALE' where deptcode='6569'

Database Constraints

- Default
  - The DEFAULT constraint is used to set a default value for a column.
  - The default value will be added to all new records, if no other value is specified.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT ‘Delhi'
);
```

Database Constraints

- Unique
  - CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    UNIQUE (ID)
    );

Database Constraints

- Not Null
  - By default, a column can hold NULL values.
  - The NOT NULL constraint enforces a column to NOT accept NULL values.
  - This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
);
```

Database Constraints

- Check
  - The CHECK constraint is used to limit the value range that can be placed in a column.
  - If you define a CHECK constraint on a column it will allow only certain values for this column.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

Deleting Rows Based on Another Table

```
DELETE FROM employees
WHERE  department_id =(SELECTdepartment_id
   FROM   departments      WHERE
   department_name LIKE '%Public%');
```

☞1 row deleted.

Deleting Rows: Integrity Constraint Error

```
DELETE FROM
   departments
WHERE      department_id
   = 60;
```

You cannot delete a row that contains a primary key that is used as a foreign key in another table

```
DELETE FROM departments
            *
ERROR at line 1:
ORA-02292: integrity constraint (HR.EMP_DEPT_FK)
violated - child record found
```

TRUNCATE

66 **Remove all tuples from table Salary**

```
TRUNCATE TABLE Salary;
```