BASIC DATABASES

Simple Queries in SQL

NGUYEN Hoang Ha

Email: nguyen-hoang.ha@usth.edu.vn

Objectives

- Understand the how to write simple queries
- Understand about NULL values
- Understand about the string data type, date-time data type
- Know how to order the output

SQL OVERVIEW

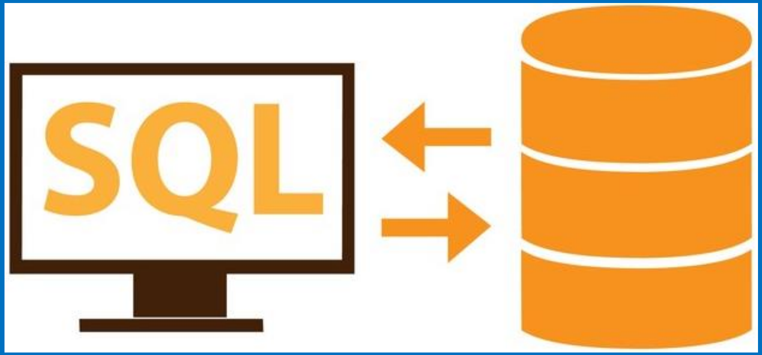
Mathematics to Computer: RA to SQL

Math

π	σ	ρ	$\bowtie_c$

RA is the conceptual basis for DB systems

Computer



What is SQL?

- SEQUEL (Structured English QUEry Language) was developed by IBM in 1974,

  later became Structural Query Language (SQL)
- Standard language to work with RDBMS
- Easy to learn
  - Close to English
  - Less than 100 words
- Pronounced as “S-Q-L” or “Sequel.”

Microsoft® SQL Server®
Linux
IBM® DB2®
Microsoft Windows®
android
UNIX®
MySQL®
iOS
Mac
ORACLE®

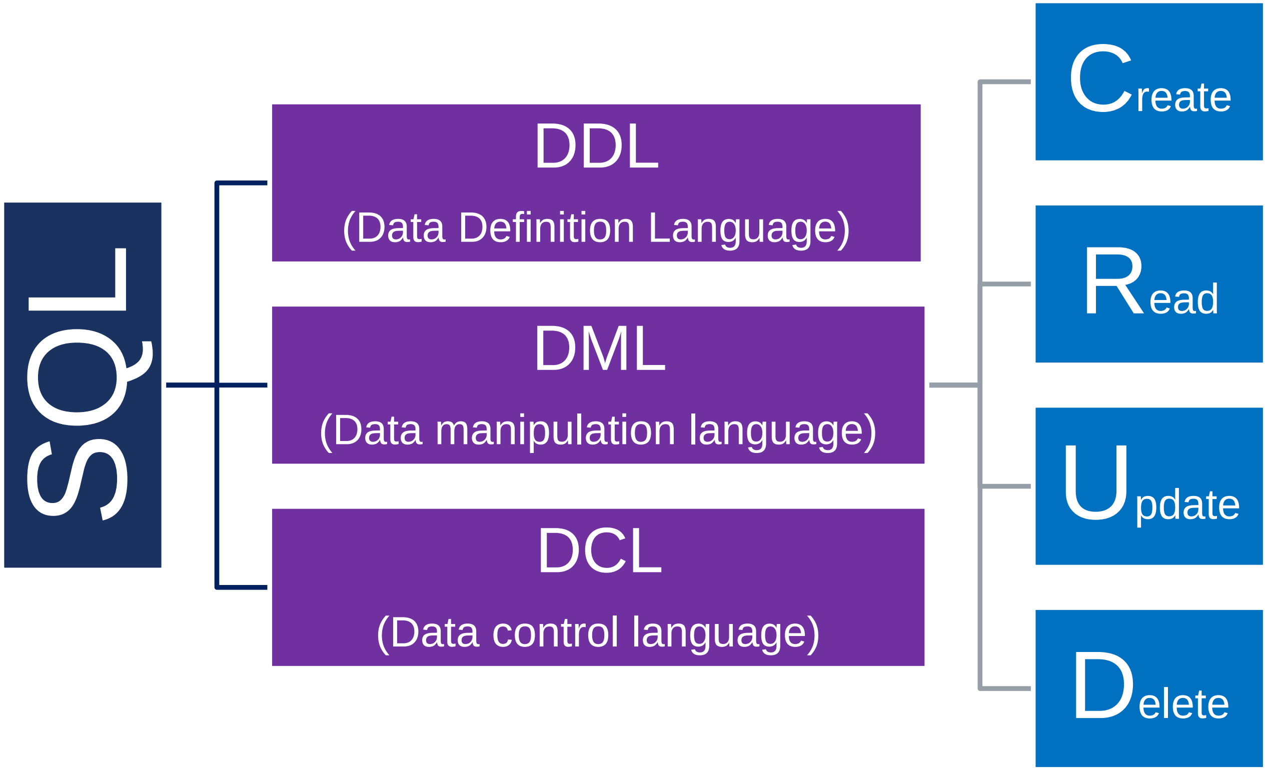
From RA to SQL

- *Based* on relational algebra, but not entirely identical.
  - Relations ⇔ Tables
  - Tuples ⇔ Rows
  - Attributes ⇔ Columns
- Unlike a relation, a table is *not* a set. Duplicates are not automatically removed.
  - This is for practical reasons. Duplicate eliminations are inefficient in implementation.
- Like a relation, the order of rows in a table is irrelevant.

SQL Revisions

Year	Name	Alias	Comments
1986	SQL-86	SQL-87	First published by ANSI. Ratified by ISO in 1987.
1989	SQL-89	FIPS 127-1	Minor revision, adopted as FIPS 127-1.
1992	SQL-92	SQL2, FIPS 127-2	Major revision (ISO 9075), *Entry Level* SQL-92 adopted as FIPS 127-2.
1999	SQL:1999	SQL3	Added regular expression matching, recursive queries, triggers, support for procedural and control-of-flow statements, non-scalar types, and some object-oriented features.
2003	SQL:2003		Introduced XML-related features, *window functions*, standardized sequences, and columns with auto-generated values (including identity-columns).
2006	SQL:2006		ISO/IEC 9075-14:2006 defines ways in which SQL can be used in conjunction with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database and publishing both XML and conventional SQL-data in XML form. In addition, it provides facilities that permit applications to integrate into their SQL code the use of XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.
2008	SQL:2008		Defines more flexible windowing functions, clarifies SQL 2003 items that were still unclear [1]

Sub-languages of SQL



SQL Commands Are Sequential

- Commands are executed in the order they are encountered.
- DDL commands are *not* like C/Java declarations.
- DDL and DML commands can be mixed
  - For example, you can define a table, fill it up with contents, and delete a columns.
  - That is, table definitions (relation schema) can be changed during the lifespan of a database.
    - The ability of doing so does imply it is a good practice.
    - It is best the schema/design of a database is well thought through before its use.

Notes

- Terminologies
  - A ***keyword*** refers to an individual SQL element.
    E.g.: SELECT and FROM are SQL elements
  - A ***clause*** is a part of a SQL statement.
    E.g.: SELECT EmployeeID, LastName is a statement
  - A ***statement*** is a combination of two or more clauses.
    E.g.: SELECT * FROM Employees is a SQL statement.
- SQL is case insensitive
- Convention: Keywords are all in UPPER CASE

DDL

DDL Commands

CRATE DATABASE

CREATE TABLE

ALTER TABLE

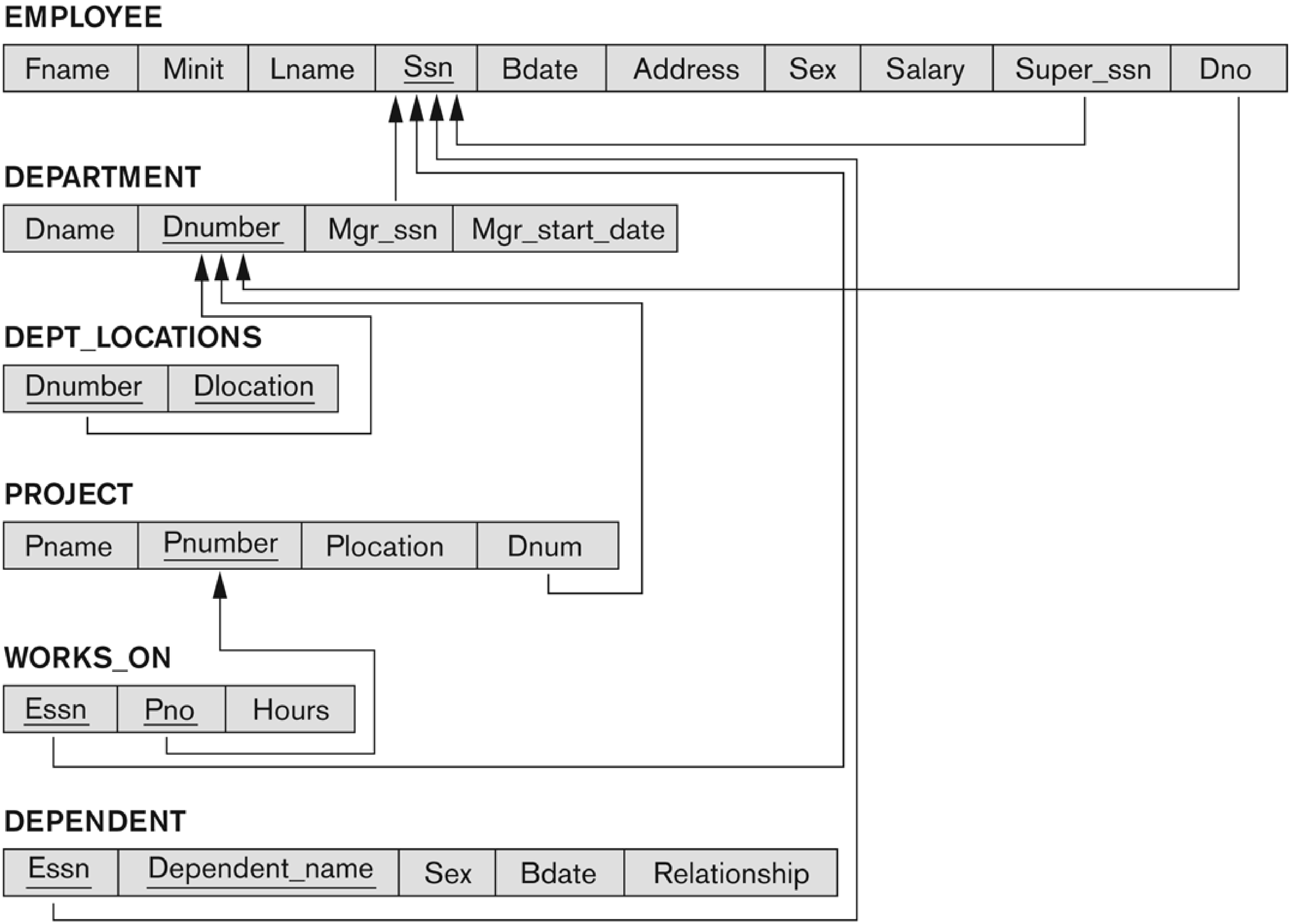
RENAME TABLE

DROP TABLE

CREATE INDEX

DROP INDEX

Also – **CREATE VIEW**



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Create Database Example

- To create

```
CREATE DATABASE Company;
```

- To use (or switch to) the database

```
USE Company;
```

- Subsequent commands will operate on the COMPANY database by default.

CREATE TABLE

```
CREATE TABLE base-table-name (colname
  datatype [column constraints – NULL/NOT
  NULL, DEFAULT…, UNIQUE,   CHECK…, PRIMARY
  KEY],

  [,colname datetype [column constraints
  …]]

  ...

 [table constraints – PRIMARY KEY…, FOREIGN
  KEY…, UNIQUE…, CHECK…]

  [storage specifications]);
```

Datatypes

- Each column must have a datatype specified
- Standards include various numeric types, fixed-length and varying-length character strings, bit strings, and user-defined types
- Available datatypes vary from DBMS to DBMS

Datatypes

- **char(n).** Fixed length character string, with user-specified length *n.*
- **varchar(n).** Variable length character strings, with user-specified maximum length *n.*
- **int.** Integer (a finite subset of the integers that is machine-dependent).
- **smallint.** Small integer (a machine-dependent subset of the integer domain type).
- **numeric(p,d).** Fixed point number, with user-specified precision of *p* digits, with *d* digits to the right of decimal point. (ex., **numeric**(3,1), allows 44.5 to be stores exactly, but not 444.5 or 0.32)
- **real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
- **float(n).** Floating point number, with user-specified precision of at least *n* digits.
- **Date**: Made up of year-month-day in the format yyyy-mm-dd
- **Time**: Made up of hour:minute:second in the format hh:mm:ss
- **Timestamp:** Has both DATE and TIME components
- **Others:** Boolean, Float, Double Precision
- See user’s manual for more data types.

CREATE TABLE Example

```
CREATE TABLE Department (
    Dname           VARCHAR(10) NOT NULL,
    Dnumber    INTEGER    Default 0,
    Mgr_ssn    CHAR(9),
    Mgr_Sartdate CHAR(9),
    PRIMARY KEY  (Dnumber),
    UNIQUE     (Dname),
    FOREIGN KEY  (Mgr_ssn) REFERENCES Employee(Ssn)
  );
```

- The “UNIQUE” clause specifies secondary keys.
- `Employee` has to be created first for the FK `Mgr_ssn` to refer to it.
- How could we have defined the `Dno` FK in `Employee`?

Adding the Dno FK to Employee

- If “`CREATE TABLE Employee`” is issued first, we cannot specify `Dno` as a FK in that `CREATE` command.
- An ALTER command must be used to change the schema of Employee, after the “`CREATE TABLE Department`,” to add a FK.

```
ALTER TABLE Employee
 ADD CONSTRAINT
    FOREIGN KEY  (Dno)
       REFERENCES Department (Dnumber);
```

The Check Clause

- Used to specify user-defined constraints
- Assume that dept. numbers are from 0 to 99.

```
CREATE TABLE Department (
    …
    Dnumber INTEGER  Default 0
        CHECK (Dnumber>=0 AND Dumber<=99),
    …);
```

- “Check” can also be a clause of the entire table.

```
CRATE TABLE Department (
    …
    Dept_create_date date,
    Mgr_start_date   date,
    CHECK (Dept_create_date <= Mgr_start_date)
  );
```

Review: Multiattribute Key

- The bar and beer together are the key for Sells:

```
CREATE TABLE Sells (
  bar    CHAR(20),
  beer   VARCHAR(20),
  price  REAL,
  PRIMARY KEY (bar, beer)
);
```

Exercise

- Create the table WORKS_ON, assuming tables EMPLOYEE and PROJECT have been created and Hours ranges from 1 to 56.

Add Columns to Existing Tables

- To add spouse SSN (`S_ssn`) to `Employee`

```
ALTER TABLE EMPLOYEE ADD COLUMN S_ssn
  char(9);
```

  - The new attribute will have NULLs in all the tuples of the relation right after the command is executed

- Alternatively, we can set a default value.

```
ALTER TABLE EMPLOYEE ADD COLUMN S_ssn
  CHAR(9) DEFAULT “000000000”;
```

Delete Columns from Existing Tables

- To delete column `S_ssn`

```
ALTER TABLE Employee DROP COLUMN S_ssn;
```

- **Reminder:** changing relation schemas typically indicates ill-executed design phase of the database.

Referential Integrity Options

- **Causes** of referential integrity violation for a foreign key FK (consider the `Mgr_ssn` of `Department`).
  - **On Delete**: when deleting the foreign tuple
    - What to do when deleting the manager tuple in `Employee` ?
  - **On Update:** when updating the foreign tuple
    - What to do when updating/changing the SSN of the manager tuple in `Employee` is changed ?
- **Actions** when the above two causes occur.
  - **Set Null**: the `Mgr_ssn` is set to null.
  - **Set Default**: the `Mgr_ssn` is set to the default value.
  - **Cascade**: the `Mgr_ssn` is updated accordingly
    - If the manager is deleted, the department is also deleted.

The Mgr_ssn Example

```
CREATE TABLE DEPARTMENT (
    …
    Mgr_ssn    CHAR(9),
    …
    FOREIGN KEY (Mgr_ssn)
       REFERENCES EMPLOYEE (Ssn)
        ON DELETE  ???
        ON UPDATE  ???
  );
```

Another Example

```
CREATE TABLE  EMP(
  …
  SSN   CHAR(9),
  DNO   INTEGER  DEFAULT 1,
  SUPERSSN  CHAR(9),
  PRIMARY KEY (ESSN),
  FOREIGN KEY (DNO) REFERENCES DEPT
      ON DELETE SET DEFAULT

         ON UPDATE  CASCADE,

   FOREIGN KEY (SUPERSSN) REFERENCES EMP
   ON DELETE SET NULL

         ON UPDATE CASCADE);
```

Miscellaneous Commands

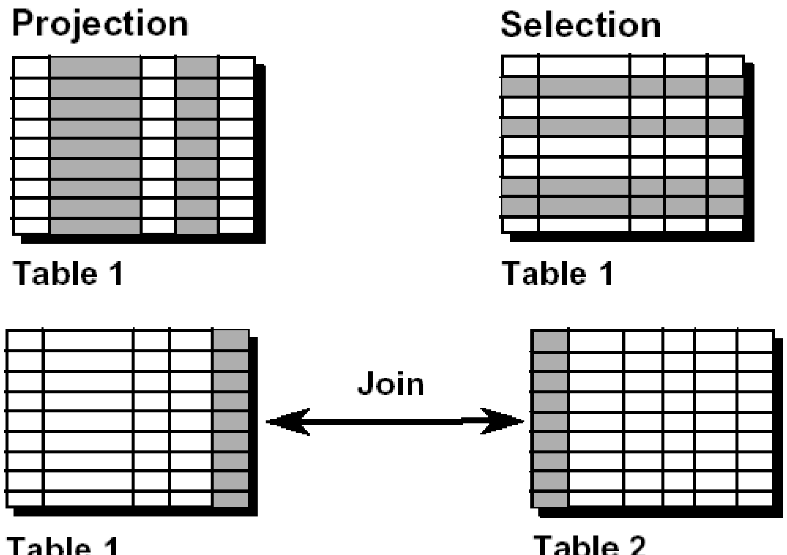
- `SHOW DATABASES;`
  - Show all the databases on the server
- `SHOW TABLES;`
  - Show all the tables of the present database
- `DROP TALBE` *`t_name`*`;`
  - Delete the entire table *t_name*
- `DROP DATABASE` *`db_name`*`;`
  - Delete the entire database *db_name*

SIMPLE DML QUERIES

SELECT commands

A SELECT statement retrieves information from the database. Using a SELECT statement, you can do the following:

- **Projection**: You can use the projection capability in SQL to choose the columns in a table that you want returned by your query.
- **Selection**: You can use the selection capability in SQL to choose the rows in a table that you want returned by a query (with WHERE clause)
- **Joining**: You can use the join capability in SQL to bring together data that is stored in different tables by creating a link between them.



SQL data retrieval query structure

SELECT desired expressions, columns π ρ

[FROM one or more tables] $\bowtie_c$

[WHERE conditions about expected rows] σ

[GROUP BY rows with the same column values]

Next lesson

[ORDER BY column list]

Syntax for a simple SELECT queries

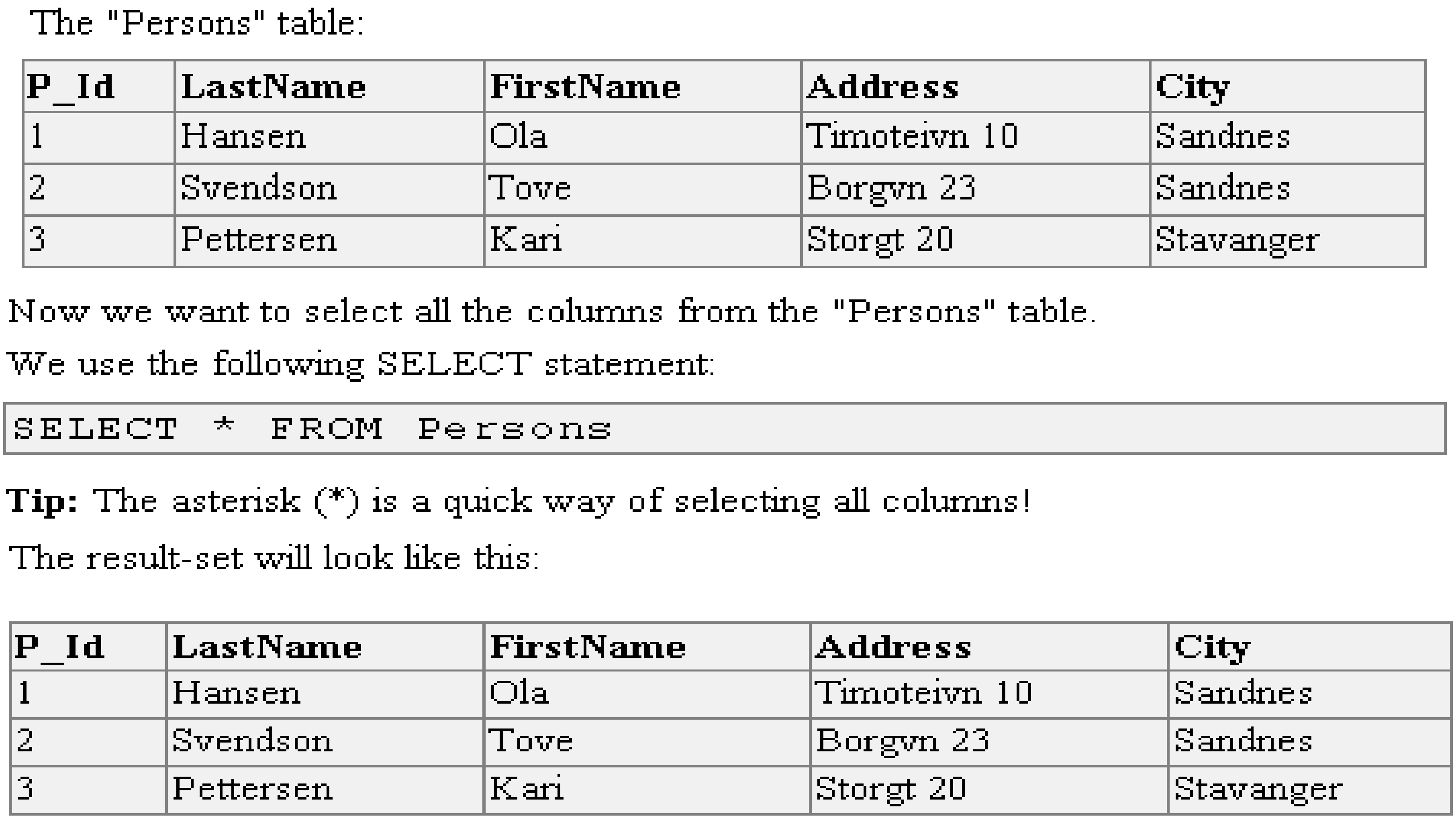
- SELECT identifies *what* columns
  - ALL: Specifies that duplicate rows can appear in the result set. ALL is the default
  - DISTINCT: Specifies that only unique rows can appear in the result set. Null values are considered equal for the purposes of the DISTINCT keyword
  - TOP *n* [PERCENT]:Specifies that only the first *n* rows are to be output from the query result set. *n* is an integer between 0 and 4294967295. If PERCENT is also specified, only the first *n* percent of the rows are output from the result set. When

    specified with PERCENT, *n* must be an integer between 0 and 100
- FROM identifies *which* table

A trick for reading & writing queries

- It’s generally easiest to examine a SELECT-FROM-WHERE query by:
  - First looking at the FROM clause to learn which relations are involved in the query
  - Then, move to the WHERE clause to learn what it is about tuples that is important to the query
  - Finally, look at the SELECT clause to see what the output format is
- The same order: FROM, then WHERE, then SELECT is often useful when writing queries of your own as well

Example: SELECT all columns



The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select all the columns from the "Persons" table.

We use the following SELECT statement:

```
SELECT * FROM Persons
```

Tip: The asterisk (*) is a quick way of selecting all columns!

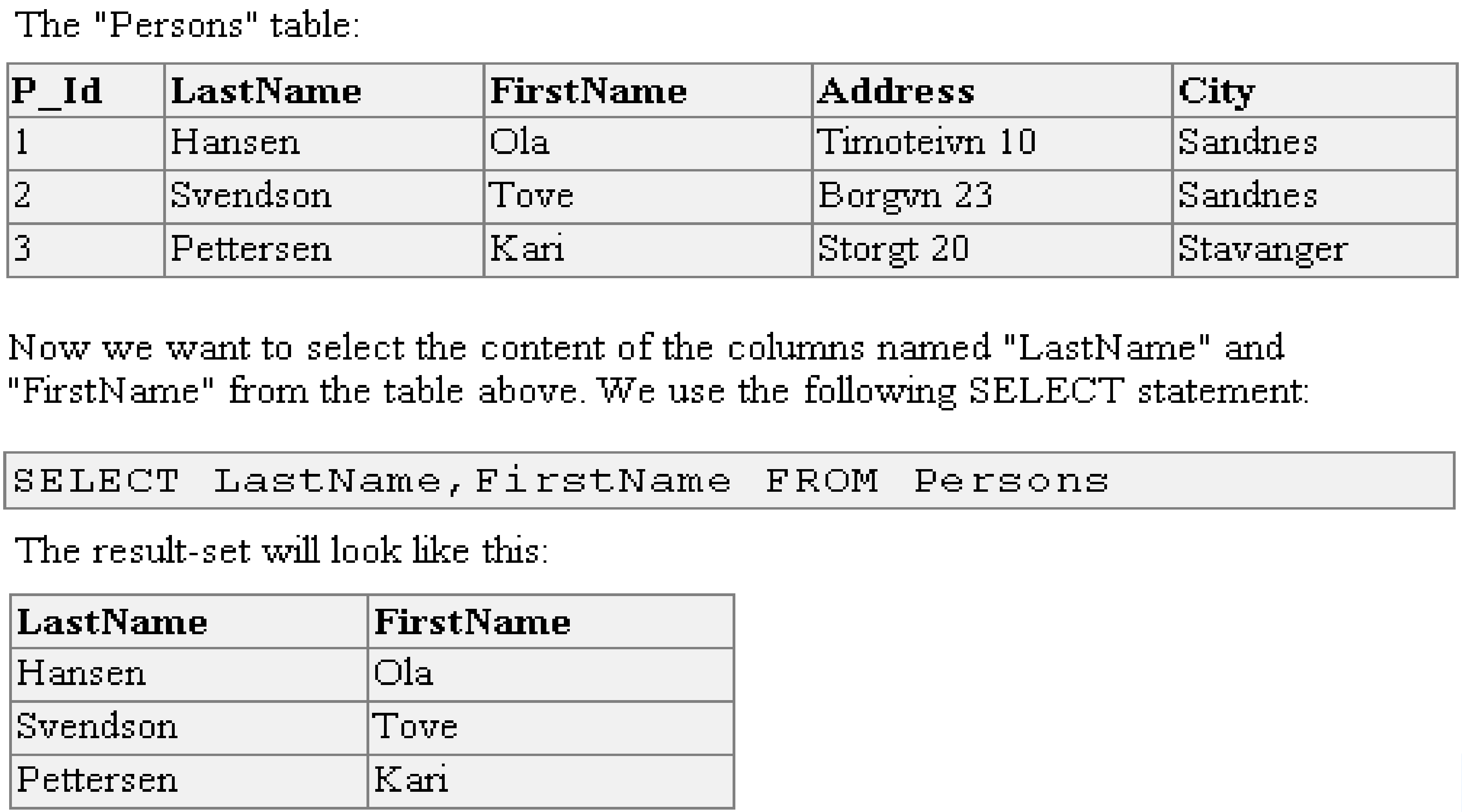
The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Projection in SQL

- We can, if we wish, eliminate some of the components of the chosen tuples; that is, we can project the relation produced by a SQL query onto some of its attributes
- In place of the * of the SELECT clause, we may list some of the attributes of the relation mentioned in the FROM clause. The result will be projected onto the attributes listed

Example: Projection in SQL



The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

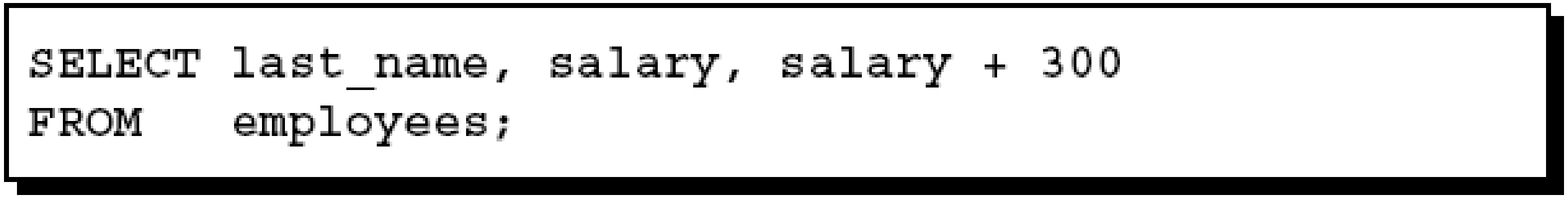
Now we want to select the content of the columns named "LastName" and "FirstName" from the table above. We use the following SELECT statement:

```
SELECT LastName,FirstName FROM Persons
```

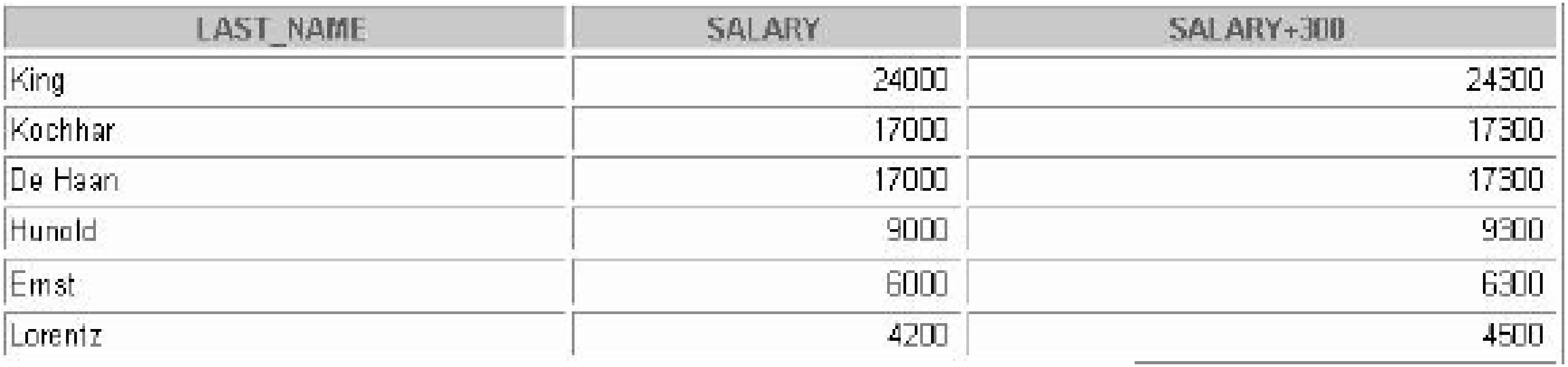
The result-set will look like this:

LastName	FirstName
Hansen	Ola
Svendson	Tove
Pettersen	Kari

Example: Extended projection using Arithmetic Operators



```
SELECT last_name, salary, salary + 300
FROM   employees;
```



LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300
Lorentz	4200	4500

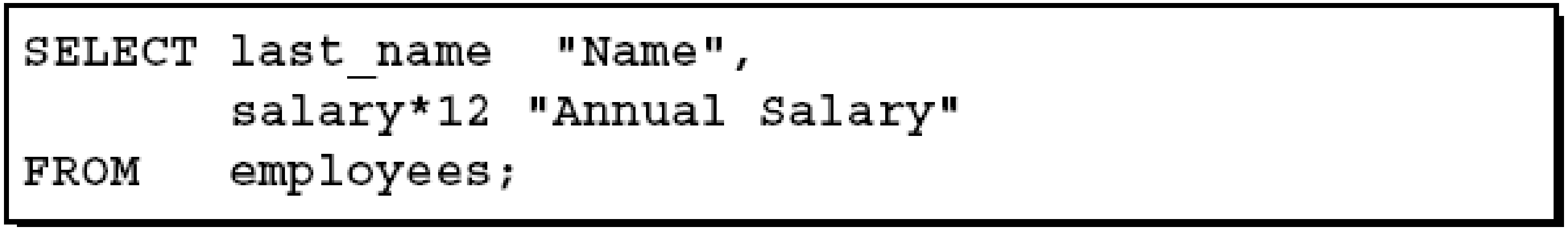
- Note that the resultant calculated column SALARY+300 is not a new column in the EMPLOYEES table; it is for display only.
- By default, the name of a new column comes from the calculation that generated it— in this case, salary+300.

Renaming or Defining a Column Alias

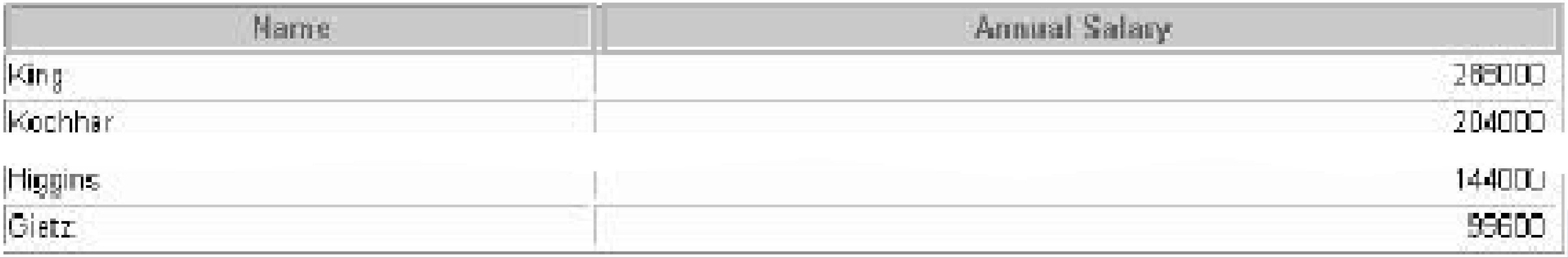
- **A column alias:**
  - Renames a column heading
  - Is useful with calculations
  - Immediately follows the column name - there can also be the optional AS keyword between the column name and alias

Example: ALIAS

- The example displays the last names and annual salaries of all the employees.
- Because Annual Salary contain a space, it has been enclosed in double quotation marks.
- Notice that the column heading in the output is exactly the same as the column alias.

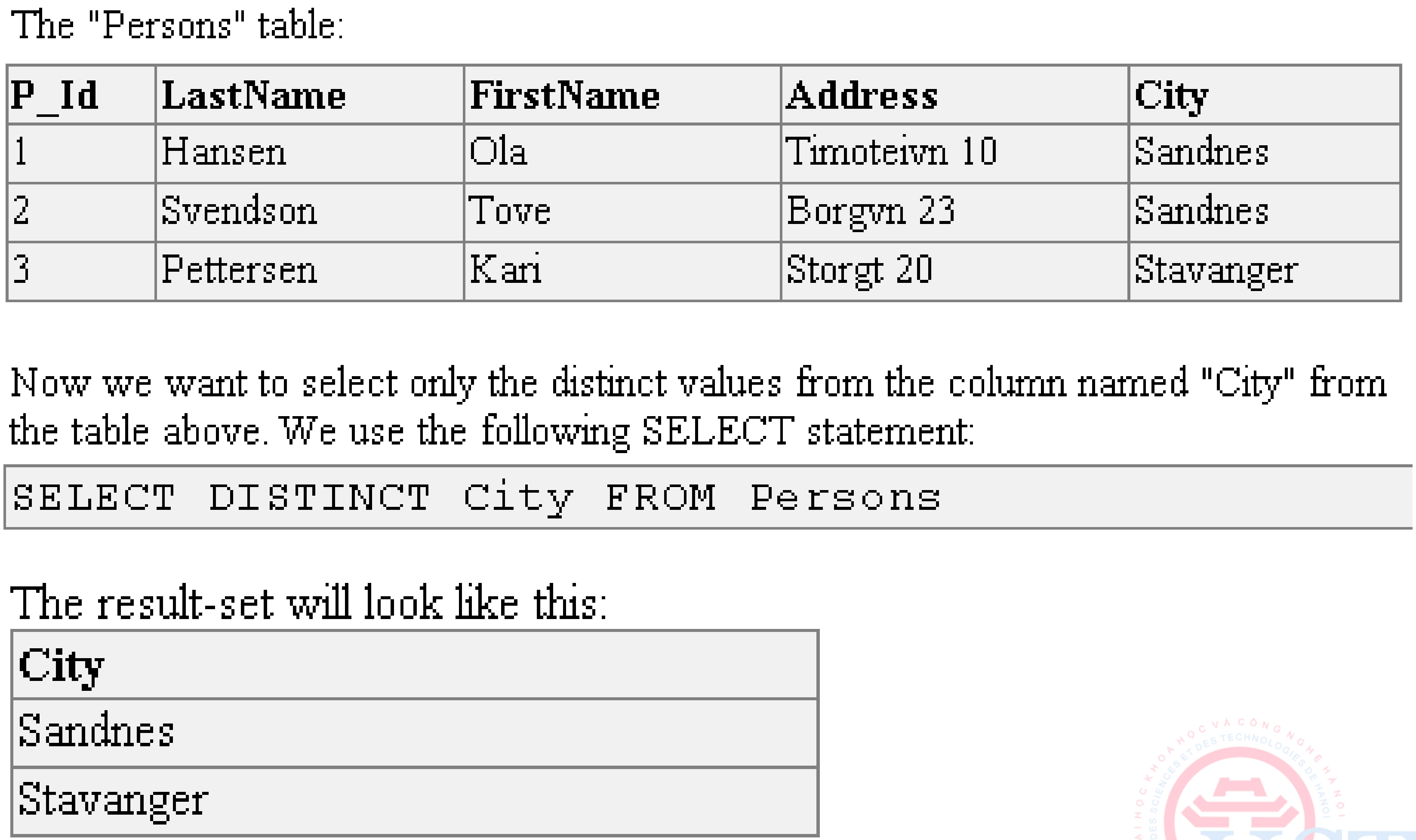


```
SELECT last_name   "Name",
       salary*12 "Annual Salary"
FROM   employees;
```



Name	Annual Salary
King	288000
Kochhar	204000
Higgins	144000
Gietz	99600

Duplication Eliminating with SELECT distinct



The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select only the distinct values from the column named "City" from the table above. We use the following SELECT statement:

```
SELECT DISTINCT City FROM Persons
```

The result-set will look like this:

City
Sandnes
Stavanger

Selection in SQL or Restricting data

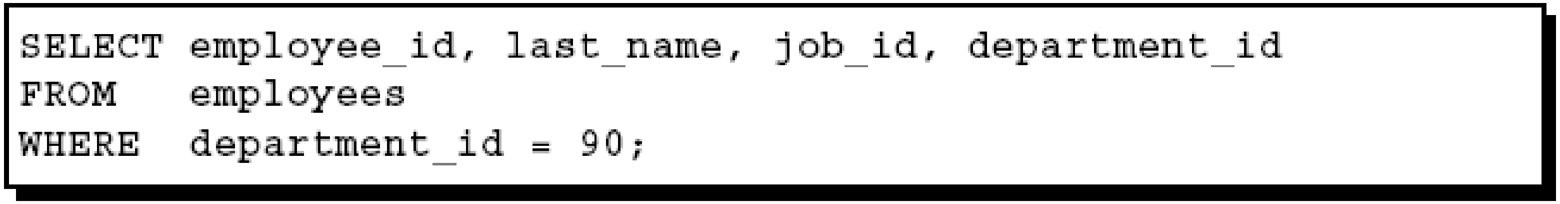
- While retrieving data from the database, you may need to restrict the rows of data that are displayed
- In that case, the solution is to use the WHERE clause
- The WHERE clause is equal to the selection operator of relational algebra
- The expression that may follow WHERE include conditional expressions like those found in C or Java

Selection in SQL (or Restricting data)

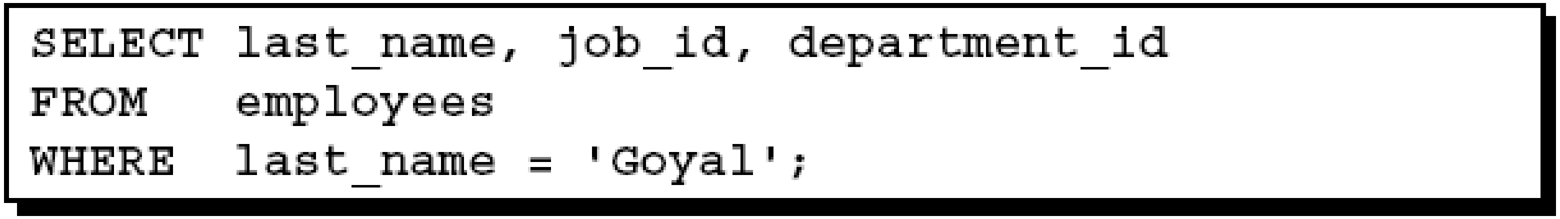
```
SELECT [ ALL | DISTINCT ]
          [ TOP n [ PERCENT ] ]
         * | {column_name | expression [alias],…}
[FROM table]
[WHERE conditions]
```

- WHERE: restricts the query to rows that meet a condition
- The WHERE clause follows the FROM clause.
- *Condition:* is composed of column names, expressions, constants, and a comparison operator

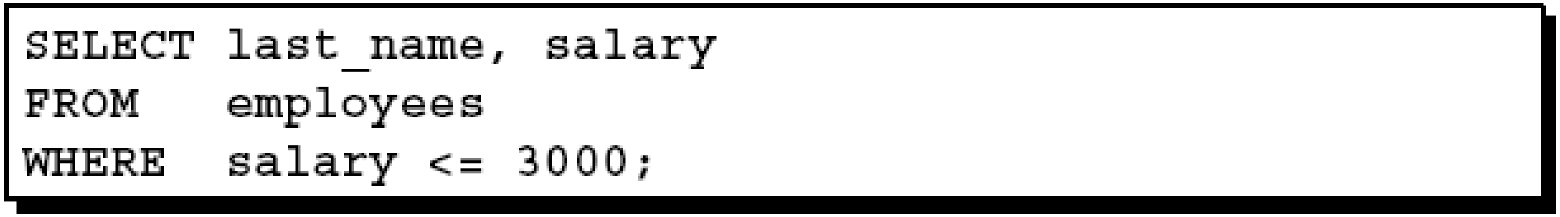
Example: Restricting data



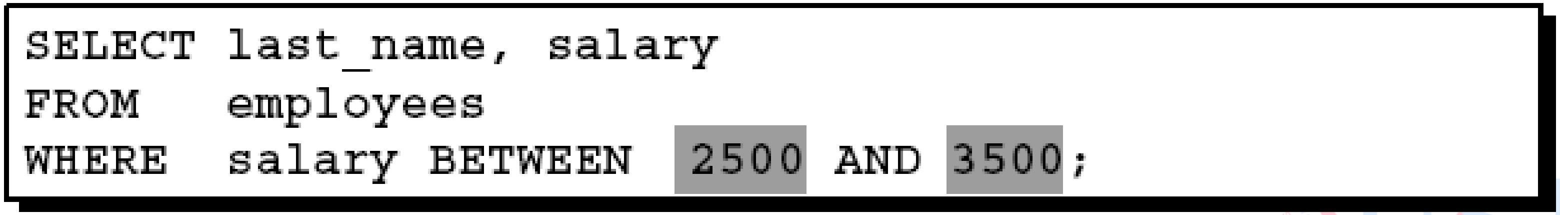
```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90;
```



```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'Goyal';
```

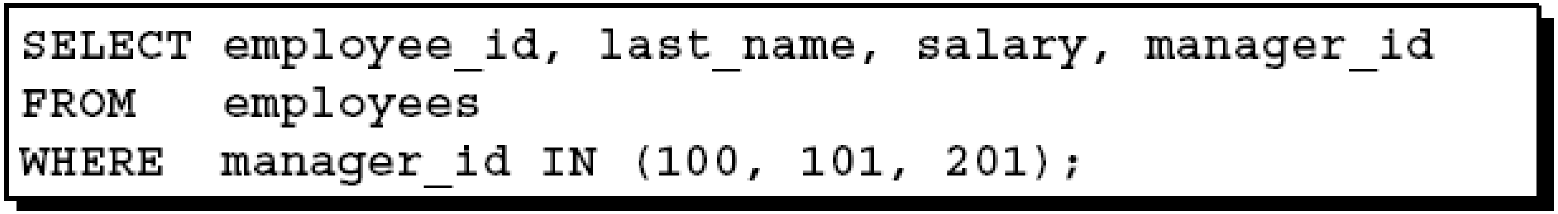


```
SELECT last_name, salary
FROM   employees
WHERE  salary <= 3000;
```

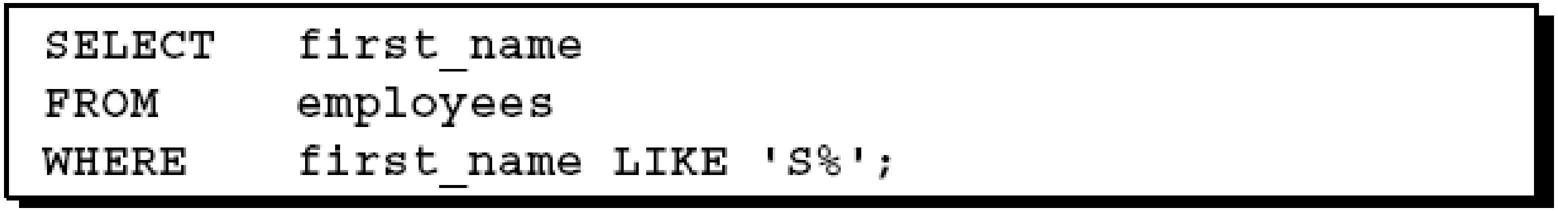


```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN  2500 AND 3500;
```

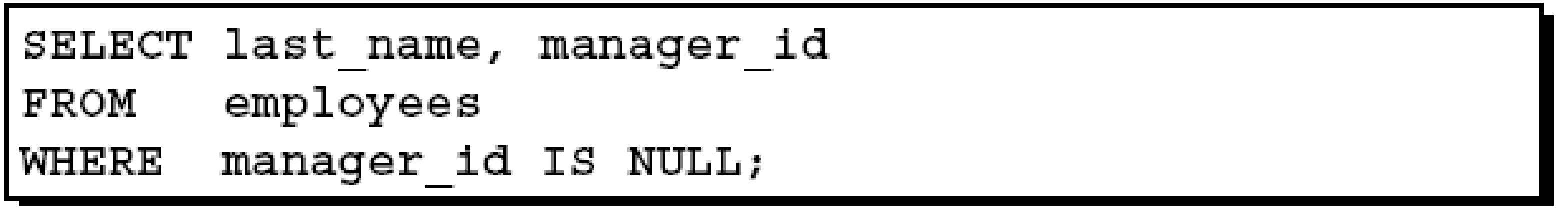
Example: Restricting data



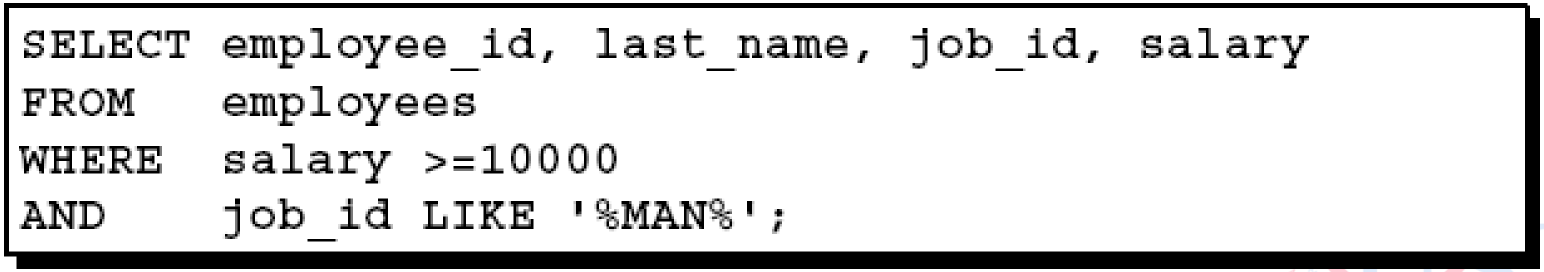
```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201);
```



```
SELECT   first_name
FROM     employees
WHERE    first_name LIKE 'S%';
```

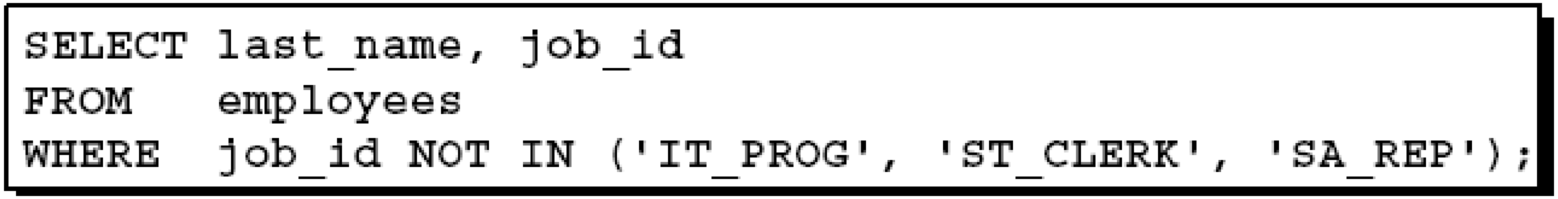


```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL;
```



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

Example: Restricting data



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

Comparison of Strings

- Two strings are equal if they are the same sequence of characters. Recall from the section 2.3.2 that strings can be stored as fixed-length strings (using CHAR) or variable-length strings (using VCHAR)
- When comparing strings with different declarations, only the actual strings are compared (SQL ignores any “pad” characters that must be presenet in the database in order to give a string its required length)
- We can use “<“, “>”, “=“ and “<>” operators to compare two strings

Pattern matching in SQL

- SQL also provides the capability to compare strings on the basis of a simple pattern match. An alternative form of comparision expression is:

  s LIKE p

  where:

  - S: is a string
  - P: is a pattern (with the optional use of some special characters: “%”, “_” ..)

- Similarly, “s NOT LIKE p” is true if and only if string s does not match pattern p

Dates and Times

- SQL generally support dates and times as special data types. These values are often representable in a variety of formats such ash:
  - ‘05/14/1948’ or
  - ’14 May 1948’
- We can compare dates or times using the same comparison operators we use for numbers or strings

NULL values

- **Null** means 'nothing' or without value or consequence
- **Null** is a special marker used in Structured Query Language (SQL) to indicate that a data value does not exist in the database. Introduced by the creator of the relational database model
- Since Null is not a member of any data domain, it is not considered a "value", but rather a marker (or placeholder) indicating the absence of value. Because of this, comparisons with Null can never result in either True or False, but always in a third logical result, Unknown
- However, certain operations on Null can return values if the value of Null is not relevant to the outcome of the operation

Ordering the Output

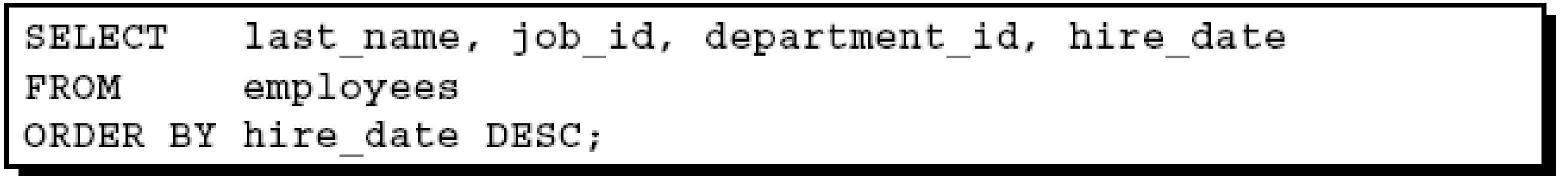
- While retrieving data from the database, you may need to specify the order in which the rows are displayed.
- In that case, the solution is to use the ORDER BY clause

Ordering the Output

- If you use the ORDER BY clause, it must be the last clause of the SQL statement.
- **Expression**: Specifies a column on which to sort.
  A sort column can be specified as a name or column alias (which can be qualified by the table or view name), an expression, or a nonnegative integer representing the position of the name, alias, or expression in select list.
- Multiple sort columns can be specified. The sequence of the sort columns in the ORDER BY clause defines the organization of the sorted result set.

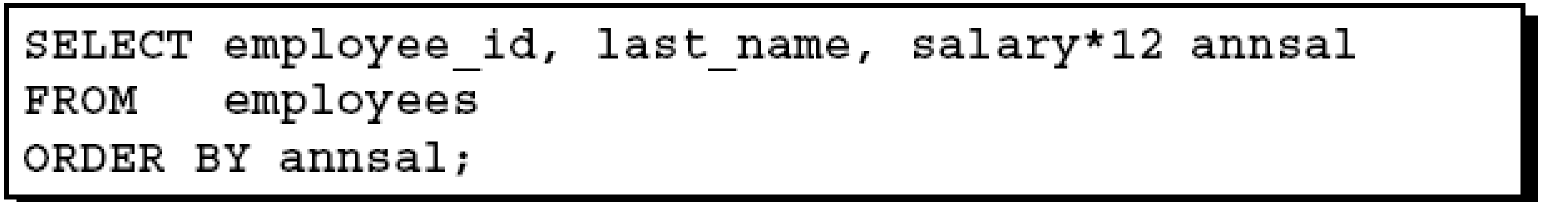
```
SELECT [ ALL | DISTINCT ]
           [ TOP n [ PERCENT ] ]
          * | {column_name | expression [alias],…}
FROM table
[WHERE conditions]
[ORDER BY {expression [ASC | DESC] ,…} ]
```

Example: Sorting data



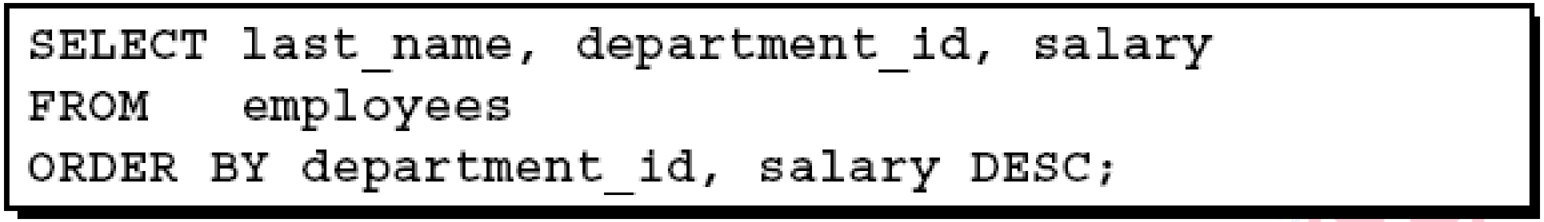
```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date DESC;
```

Sorting by Column Alias



```
SELECT employee_id, last_name, salary*12 annsal
FROM   employees
ORDER BY annsal;
```

Sorting by Multiple Columns



```
SELECT last_name, department_id, salary
FROM   employees
ORDER BY department_id, salary DESC;
```

Exercise 1

Write SQL queries to create the following tables:

- STUDIOS (name, address)
- STARS (name, address, phone)
- MOVIES (title, year, length, genre)

After creating, write SQL queries to drop them

Exercise 2

Write SQL queries to do following tasks:

- Add a column named DESCRIPTION into MOVIES table (you must determine the data type for it)
- Add a column named HOBBIES into STARS table (you must determine the data type for it)
- Add a column named BIRTHDAY into STARS table (you must determine the data type for it)

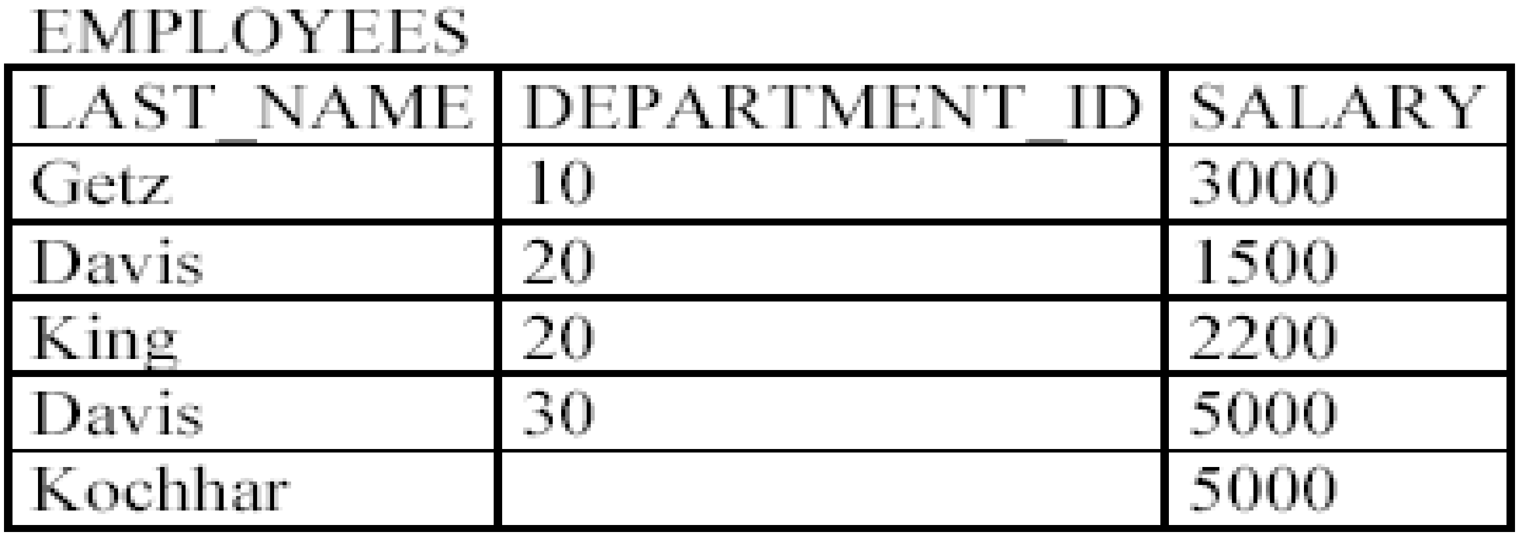
Exercise 3

Write SQL queries to do following tasks:

- Remove the column named DESCRIPTION from MOVIES table
- Remove the column named HOBBIES from STARS table
- Remove the column named BIRTHDAY from STARS table

Exercise 4

- Write a SQL query to show all tuples in table EMPLYEES

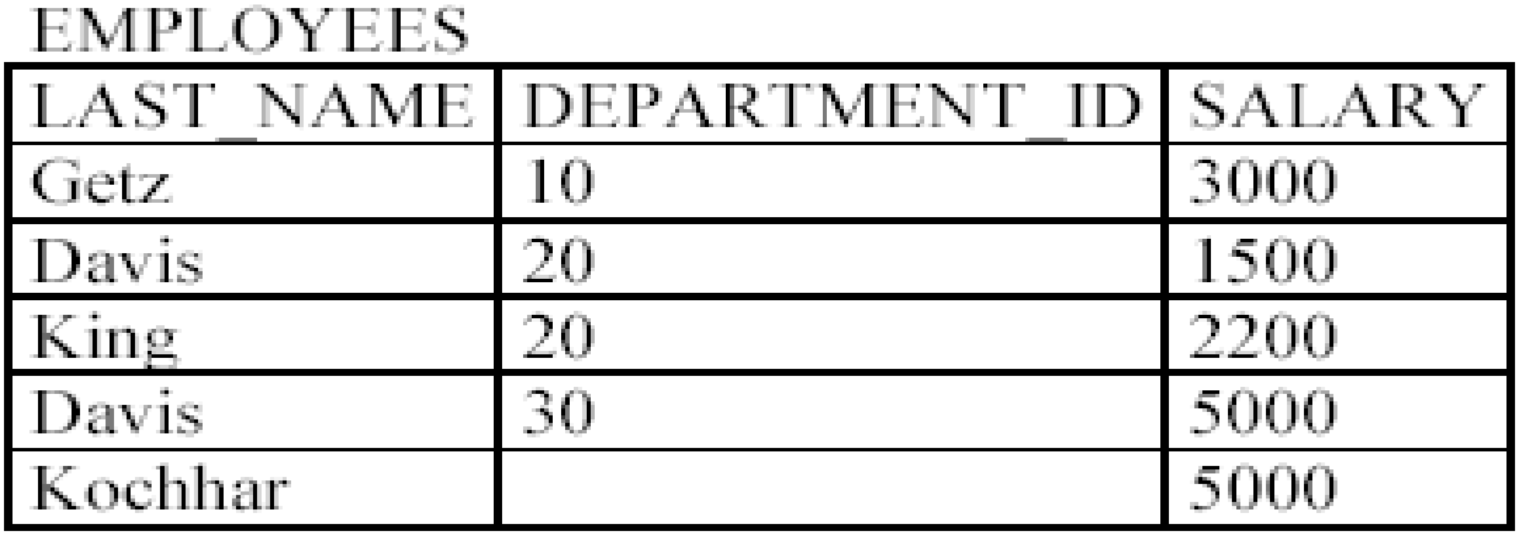


EMPLOYEES

LAST_NAME	DEPARTMENT_ID	SALARY
Getz	10	3000
Davis	20	1500
King	20	2200
Davis	30	5000
Kochhar		5000

Exercise 5

- Write a SQL query to show all SALARY (but eliminating duplications) in table EMPLYEES
- Write a SQL query to show all DEPARTMENT_ID (but eliminating duplications) in table EMPLYEES

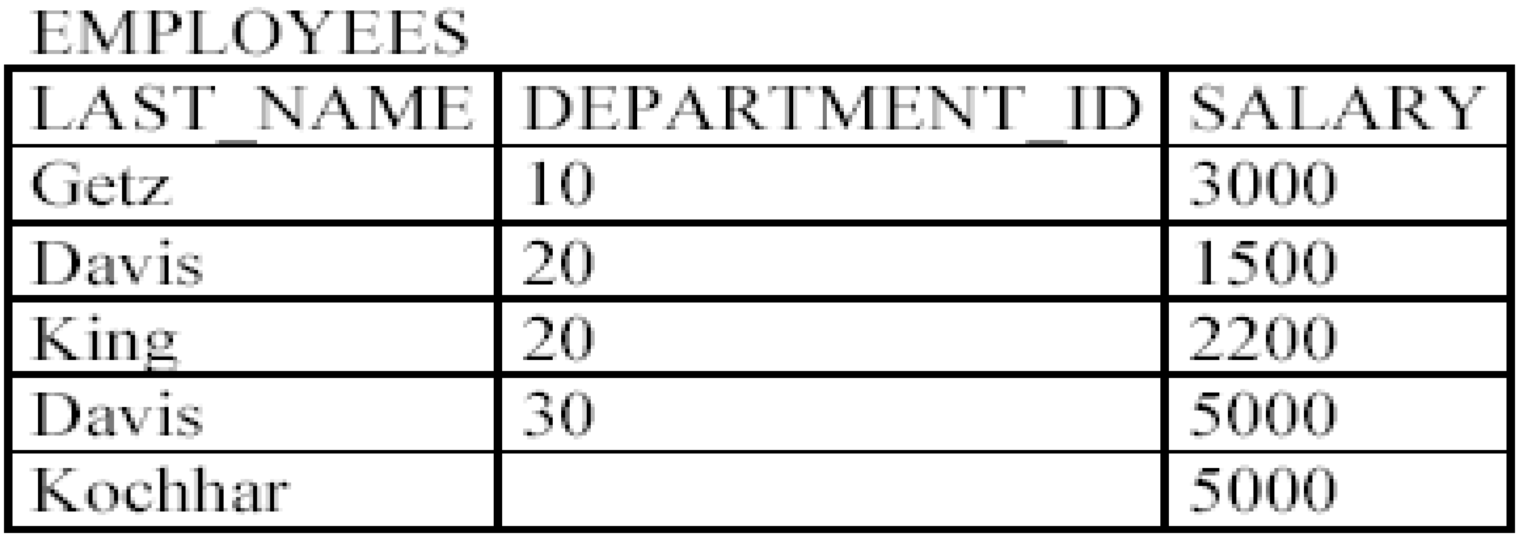


EMPLOYEES

LAST_NAME	DEPARTMENT_ID	SALARY
Getz	10	3000
Davis	20	1500
King	20	2200
Davis	30	5000
Kochhar		5000

Exercise 6

- Write a SQL query to delete all tuples in EMPLOYEES table
- Write a SQL query to delete all tuples with NULL value in DEPARTMENT_ID

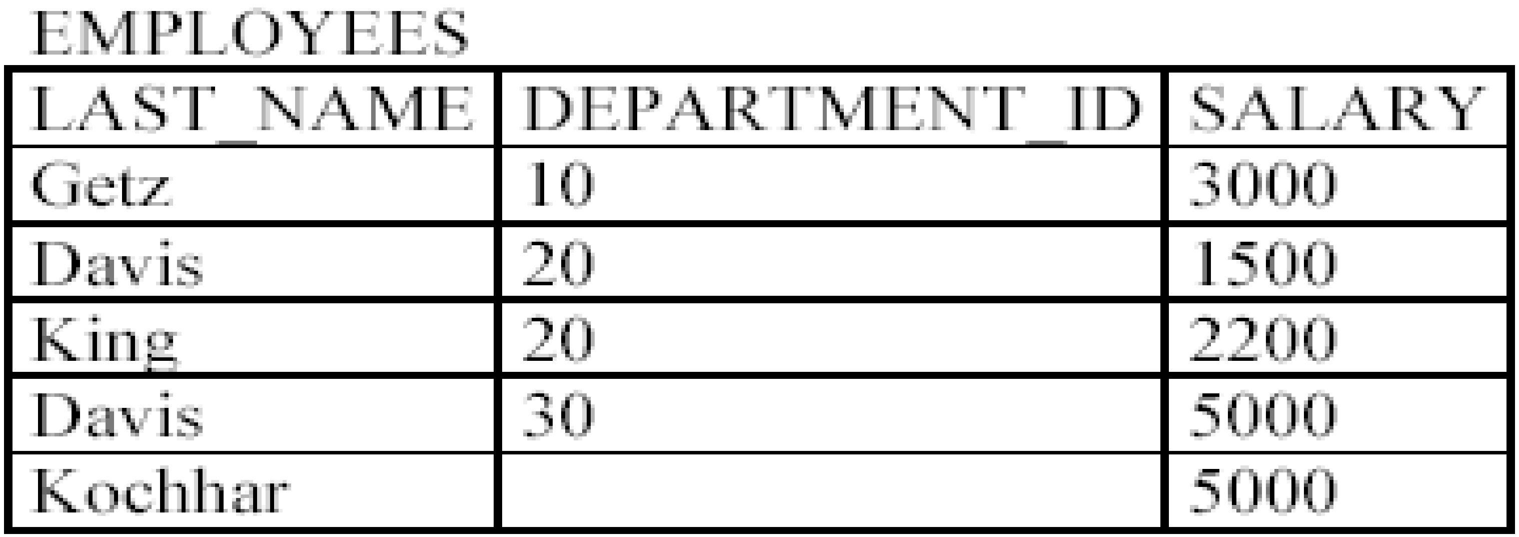


EMPLOYEES

LAST_NAME	DEPARTMENT_ID	SALARY
Getz	10	3000
Davis	20	1500
King	20	2200
Davis	30	5000
Kochhar		5000

Exercise 7

- Write a SQL query to set DEPARTMENT_ID to the value 10
- Write a SQL query to set DEPARTMENT_ID to the value 10 if DEPARTMENT_ID is NULL

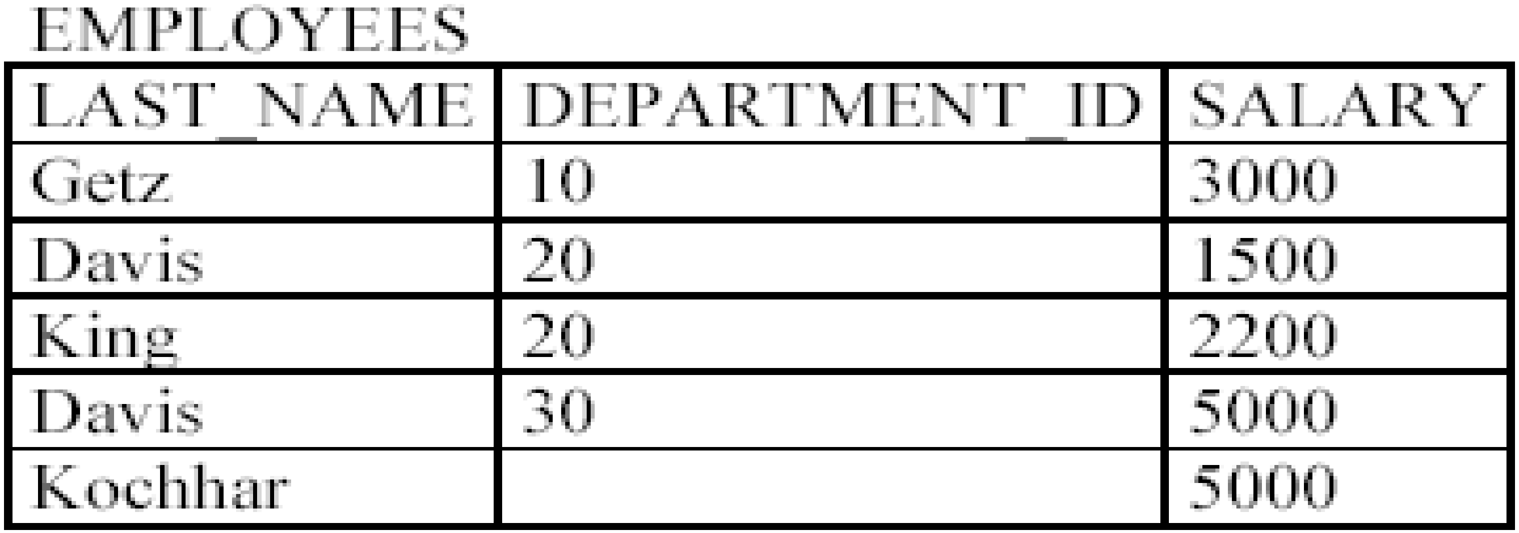


EMPLOYEES

LAST_NAME	DEPARTMENT_ID	SALARY
Getz	10	3000
Davis	20	1500
King	20	2200
Davis	30	5000
Kochhar		5000

Exercise 8

- Write a SQL query to insert some new tuples into EMPLOYEES



EMPLOYEES

LAST_NAME	DEPARTMENT_ID	SALARY
Getz	10	3000
Davis	20	1500
King	20	2200
Davis	30	5000
Kochhar		5000