SQL: basic queries

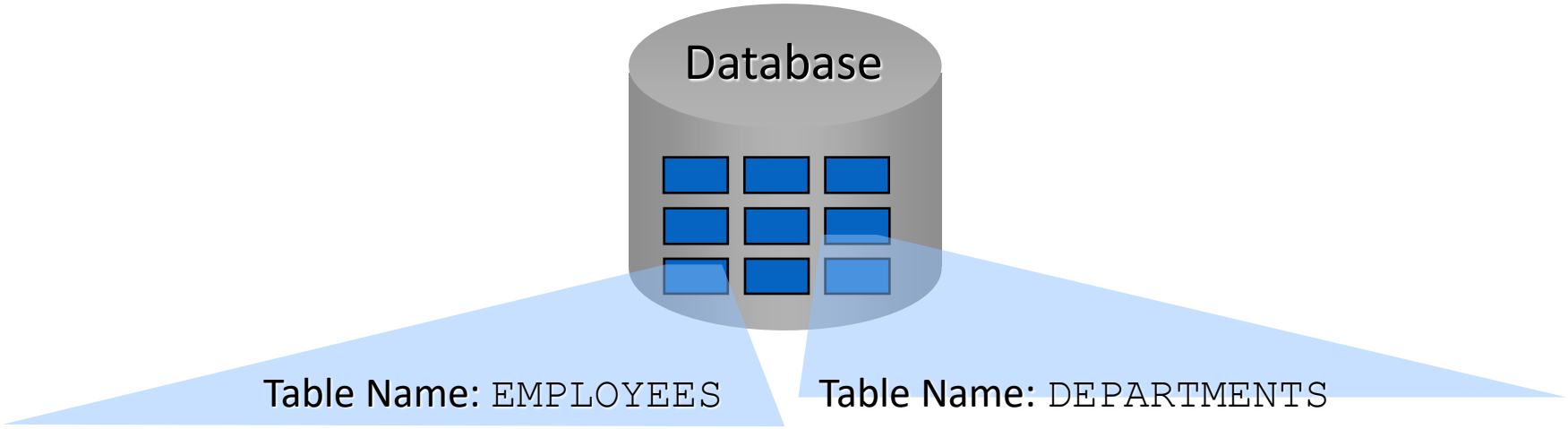
Agenda

- Short revision on relational databases
- Short revision on SQL and stored programs
- Basic SELECT queries

Short revision on relational databases

Short Revision on Relational Databases

- A **relational database** is a collection of tables and relationships between them

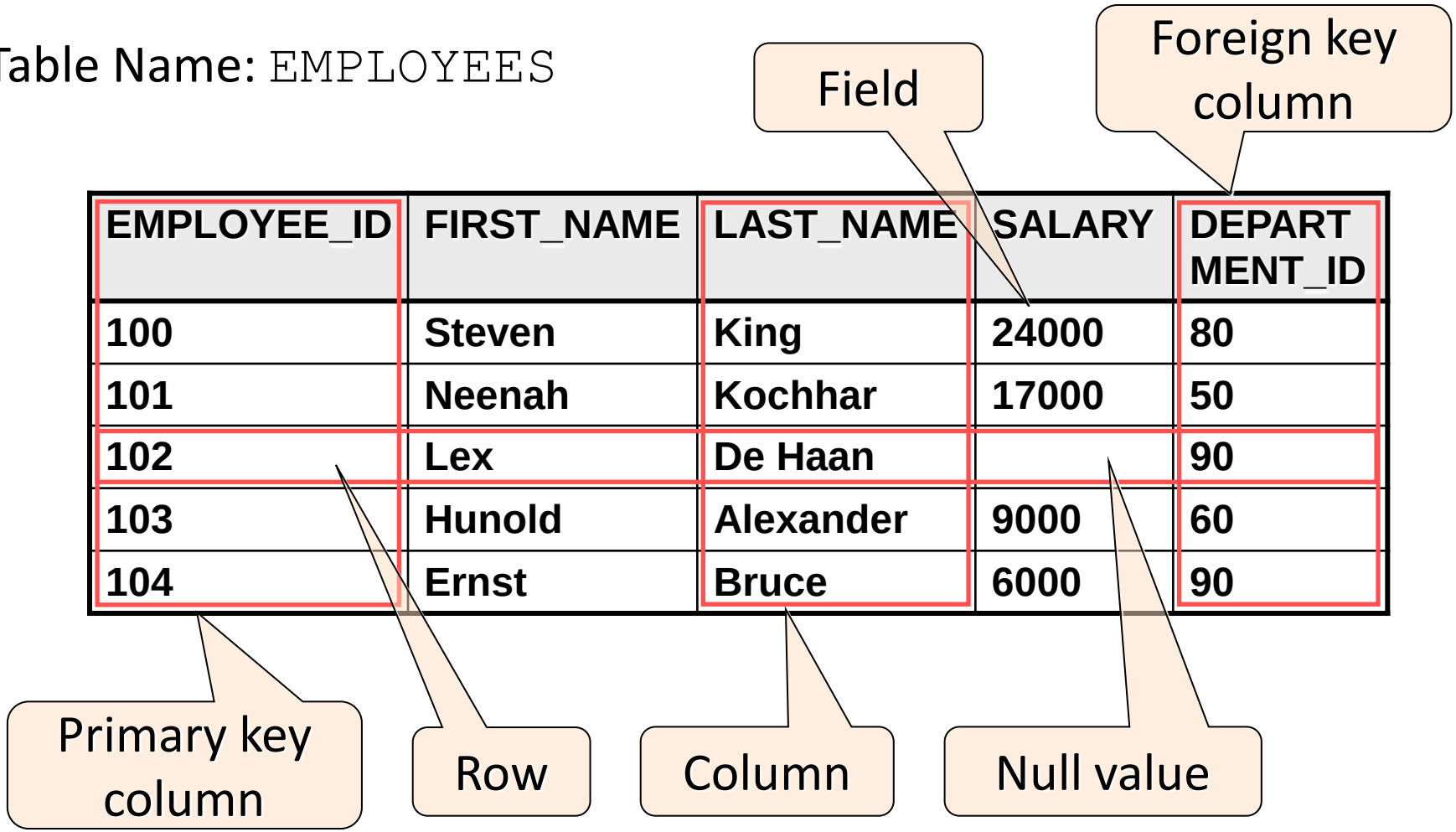


EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL
100	Steven	King	SKING
101	Neenah	Kochhar	NKOCH
102	Lex	De Haan	HAAN

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID
10	Administration	200
20	Marketing	201
50	Shipping	124

Short Revision on Relational Databases

Table Name: `EMPLOYEES`



Short Revision on Relational Databases

- Each row of data in a table is uniquely identified by a primary key (PK)
- You can logically relate data from multiple tables using foreign keys (FK)

Table Name: EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	80
101	Neenah	Kochhar	50
102	Lex	De Haan	90

Table Name: DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
50	Shipping

Primary key

Foreign key

Primary key

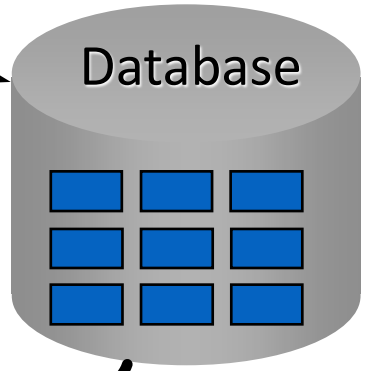
Short revision on SQL and stored procedures

Short Revision on SQL and Stored Procedures

SQL statement is entered

```
SELECT NAME
FROM DEPARTMENTS
```

SQL statement is sent to the database



NAME
Administration
Marketing
Shipping

The result is returned (usually as a table)

Short Revision on SQL and Stored Procedures

- Structured Query Language (SQL)
  - Declarative language for query and manipulation of relational data

- SQL consists of:
  - Data Manipulation Language (DML) - SELECT, INSERT, UPDATE, DELETE
  - Data Definition Language (DDL) - CREATE, DROP, ALTER, GRANT, REVOKE

- Sample SQL command:

```
SELECT * FROM USERS
```

Short Revision on SQL and Stored Procedures

- MySQL stored programs are an extension to the standard SQL language:
  - Provide if statements, loops, exceptions - like the high-level procedural programming languages
  - Used for writing procedures, functions, triggers, etc.

Short Revision on SQL and Stored Procedures

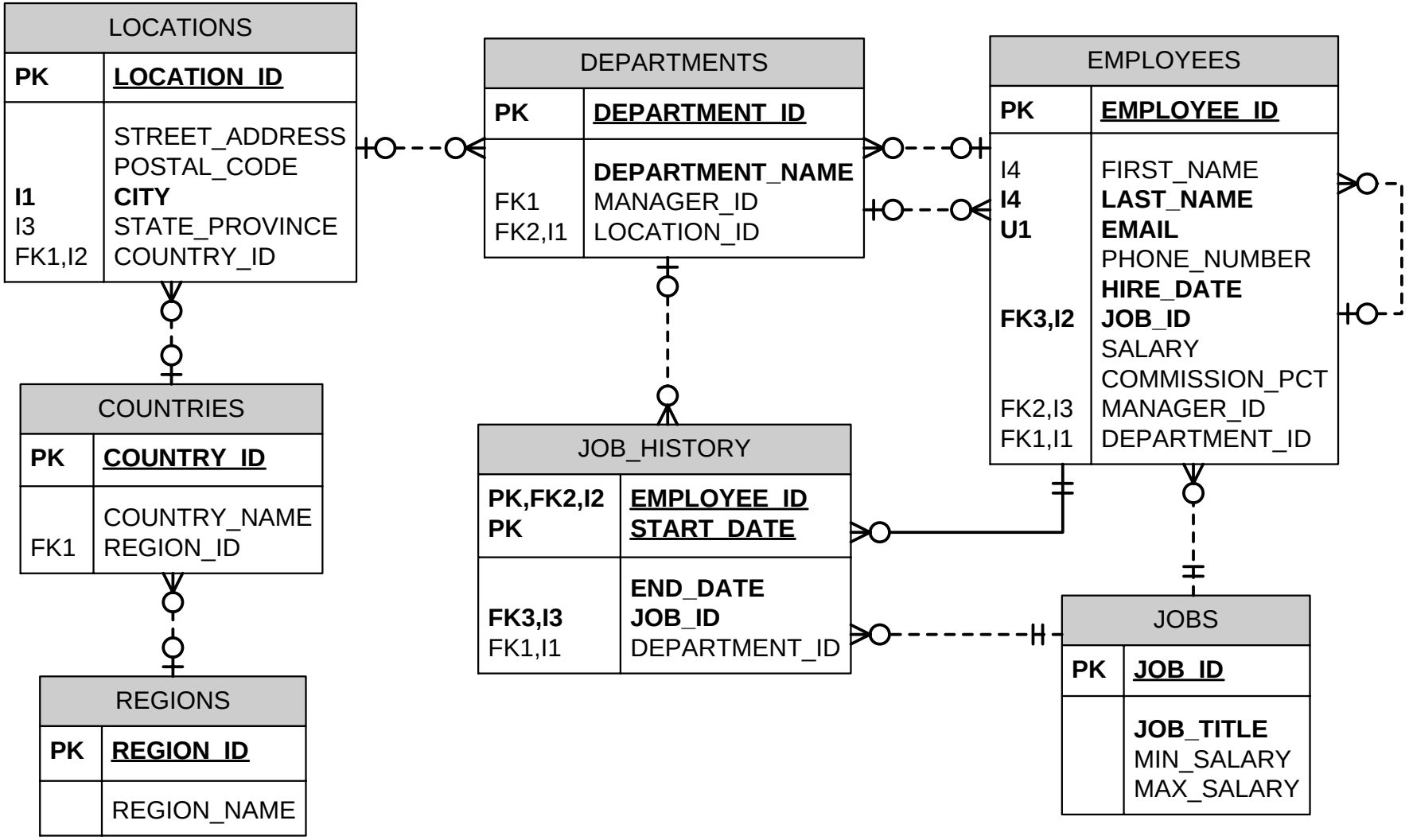
- Example stored program:

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `emp_total_salaries`()
BEGIN
    DECLARE p_name VARCHAR(100);
    DECLARE p_salary INT;
    DECLARE TOTAL_SALARIES INT DEFAULT 0;
    DECLARE FINISHED INTEGER DEFAULT 0;
    DECLARE EMP_CURSOR CURSOR FOR
        SELECT NAME, SALARY from EMPLOYEES;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
OPEN EMP_CURSOR;
GET_EMP: LOOP
     FETCH EMP_CURSOR INTO p_name, p_salary;
     IF FINISHED = 1 THEN LEAVE GET_EMP; END IF;
     SET TOTAL_SALARIES =  TOTAL_SALARIES + p_salary;
END LOOP GET_EMP;
CLOSE EMP_CURSOR;
SELECT (CONCAT('Total salaries: ' , TOTAL_SALARIES));
END
```

Alternative HRM Database Schema

Diagram

Short Revision on SQL and Stored Procedures



Basic SQL Statements

Introducing the `SELECT` Statement

Basic SELECT queries

Capabilities of SQL `SELECT`

Projection

Take some of the columns

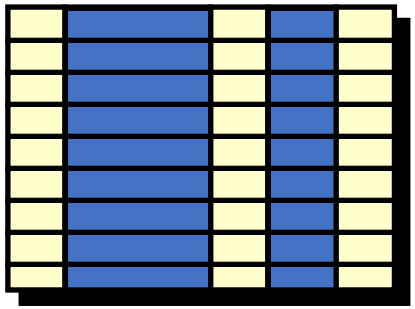


Table 1

Selection

Take some of the rows

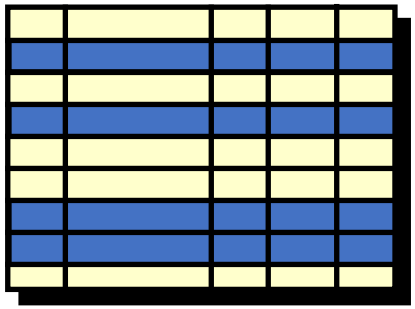


Table 1

Join

Combine tables by some column

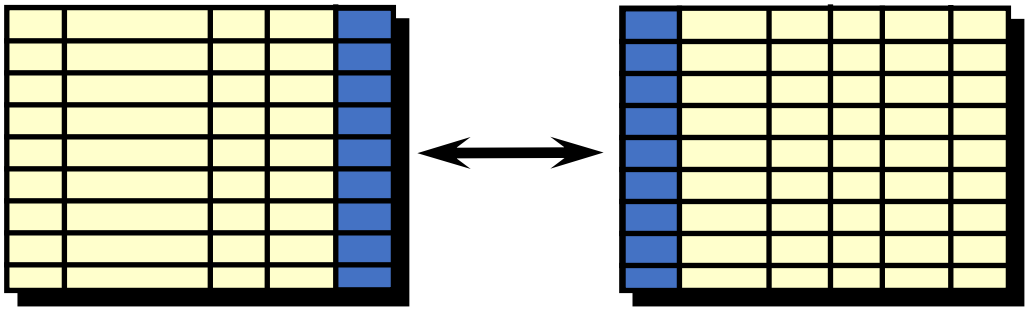


Table 1

Table 2

Basic SELECT queries

- Basic `SELECT` Statement:

```
SELECT *|{[DISTINCT] column|expression
[alias],...}
FROM table
```

- SELECT identifies the columns
- FROM identifies the table

Basic SELECT queries

- Example: Selecting all departments

```
SELECT * FROM DEPARTMENTS
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1900

- Example: Selecting specific columns

```
SELECT
  DEPARTMENT_ID,
  LOCATION_ID
FROM DEPARTMENTS
```

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1900

Basic SELECT queries

- Arithmetic operators can be used in queries:
  - +, -, *, /
- Example:

```
SELECT LAST_NAME, SALARY, SALARY + 300
FROM EMPLOYEES
```

LAST_NAME	SALARY	SALARY + 300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300

Basic SELECT queries

- A null is a value that is unavailable, unassigned, unknown, or inapplicable
  - Not the same as zero or a blank space
- Arithmetic expressions containing a `null` value are evaluated to `null`

```
SELECT LAST_NAME, MANAGER_ID FROM EMPLOYEES
```

LAST_NAME	MANAGER_ID
King	(null)
Kochhar	100
De Haan	100

`NULL` is displayed as empty space or as (null)

Basic SELECT queries

- Select without FROM causes an error:

```
SELECT 2+3
```

- Oracle supports special pseudo-table called `DUAL`
  - Selecting from `DUAL` always returns a single row - for example (Oracle only):

```
SELECT 'Ivan' || ' ' || 'Ivanov' FROM DUAL
```

```
SELECT '2*3=' || (2*3) FROM DUAL
```

Basic SELECT queries

- A column aliases specifies an alternative name for a column - the optional keyword AS is used
- An alias follows the column name
- Double quotation must be used marks if alias contains spaces

```
SELECT LAST_NAME "Name", 12*SALARY AS
  "Annual Salary" FROM EMPLOYEES
```

Name	Annual Salary
King	288000
Kochhar	204000

Basic SELECT queries

- The concatenation operator in Oracle is used to concatenate columns or character strings to other columns
- It is represented by two vertical bars (| |)
- Creates a resultant column that is the result of a character expression - for example (Oracle only)

```
SELECT LAST_NAME || JOB_ID AS "Employees"
FROM EMPLOYEES
```

Employees
KingAD_PRES
KochharAD_VP
De HaanAD_VP

Basic SELECT queries

- A literal is a character, a number, or a date included in the `SELECT` list
- Date and character literal values must be enclosed within single quotation marks

```
SELECT CONCAT(LAST_NAME, ' is a ', JOB_ID) AS
"Employee Details" FROM EMPLOYEES
```

Employees
King is a AD_PRES
Kochhar is a AD_VP
De Haan is a AD_VP

Basic SELECT queries

- The default display of queries is all rows, including duplicate rows

```
SELECT DEPARTMENT_ID
FROM EMPLOYEES
```

DEPARTMENT_ID
90
90
60
...

- duplicate rows are eliminated by using the `DISTINCT` keyword in the `SELECT` clause

```
SELECT
  DISTINCT DEPARTMENT_ID
FROM EMPLOYEES
```

DEPARTMENT_ID
90
60
...

Basic SELECT queries

- `UNION` combines the results from several `SELECT` statements
  - The columns count, types and order should match

```
SELECT FIRST_NAME AS NAME
FROM EMPLOYEES
UNION
SELECT LAST_NAME AS NAME
FROM EMPLOYEES
```

NAME
Abel
Adam
Alana
...

- `INTERSECT` / `MINUS` perform logical intersection / difference between given two sets of records

Basic SELECT queries

- The number of rows can be restricted by using the `WHERE` clause:

```
SELECT LAST_NAME,
DEPARTMENT_ID FROM
EMPLOYEES WHERE
DEPARTMENT_ID = 90
```

LAST_NAME	DEPARTMENT_ID
King	90
Kochhar	90
De Haan	90

- More examples:

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID FROM
EMPLOYEES WHERE LAST_NAME = 'Whalen'
```

```
SELECT LAST_NAME, SALARY FROM EMPLOYEES
WHERE SALARY <= 3000
```

Basic SELECT queries

- The `BETWEEN` operator is used to specify a search range:

```
SELECT LAST_NAME, SALARY FROM EMPLOYEES
WHERE SALARY BETWEEN 2500 AND 3000
```

- The `IN` / `NOT IN` operator is used to specify a set of values for search:

```
SELECT FIRST_NAME, LAST_NAME, MANAGER_ID FROM
EMPLOYEES WHERE MANAGER_ID IN (100, 101, 201)
```

Basic SELECT queries

- The `LIKE` operator to specify a search pattern:

```
SELECT FIRST_NAME FROM EMPLOYEES
WHERE FIRST_NAME LIKE 'S%'
```

  - `%` means 0 or more chars
  - `_` means one char

Basic SELECT queries

- Checking for a `NULL` value:

```
SELECT LAST_NAME, MANAGER_ID FROM EMPLOYEES
WHERE MANAGER_ID IS NULL
```

```
SELECT LAST_NAME, MANAGER_ID FROM EMPLOYEES
WHERE MANAGER_ID IS NOT NULL
```

- Attention: `COLUMN=NULL` is always false!

```
SELECT LAST_NAME, MANAGER_ID FROM EMPLOYEES
WHERE MANAGER_ID=NULL
```

This is always false!

```
SELECT LAST_NAME, MANAGER_ID FROM EMPLOYEES
WHERE NULL=NULL
```

This is always false!

Basic SELECT queries

- OR and AND logical operators can be used to combine statements:

```
SELECT LAST_NAME, JOB_ID, SALARY FROM EMPLOYEES
WHERE SALARY >= 1000 AND JOB_ID LIKE '%MAN%'
```

```
SELECT LAST_NAME FROM EMPLOYEES
WHERE COMMISSION_PCT IS NOT NULL
  OR LAST_NAME LIKE '%S%'
```

- NOT operator can be used to negate a statement:

```
SELECT LAST_NAME, SALARY, MANAGER_ID
FROM EMPLOYEES
WHERE NOT (MANAGER_ID IS NULL) AND
NOT (SALARY>10000)
```

Basic SELECT queries

- Rows can be sorted with the `ORDER BY` clause
  - `ASC`: ascending order, default
  - `DESC`: descending order

```
SELECT LAST_NAME,
HIRE_DATE FROM EMPLOYEES
ORDER BY HIRE_DATE
```

LAST_NAME	HIRE_DATE
King	1987-06-17
Whalen	1987-09-17
Kochhar	1989-09-21

```
SELECT LAST_NAME,
HIRE_DATE FROM EMPLOYEES
ORDER BY HIRE_DATE DESC,
LAST_NAME
```

LAST_NAME	HIRE_DATE
Banda	2000-04-21
Kumar	2000-04-21
Ande	2000-03-24

Questions ?

Exercises (1)

1. What is SQL? What is DML? What is DDL? Explain the most important SQL commands.
2. What is a MySQL stored procedure, function and routine ?
3. Start Oracle SQL Developer and connect to the database. Use the HRM user. Examine the major tables in the HRM schema (all exercises below use the original HRM schema for the course).
4. Write an SQL query to display all information about all departments.
5. Write an SQL query to find all department names.
6. Write an SQL query to find the salary of each employee by HIREDATE.

Exercises (2)

7. Write an SQL query to find the email addresses of each employee. Consider that the mail domain is mail.somecompany.com. Emails should look like "bernst@mail.somecompany.com". The produced column should be named "Full Email Address".
8. Write an SQL query to find all different salaries that are paid to the employees. Use `DISTINCT`.
9. Write a SQL query to find all departments and all region names, country names and city names as a single list. Use `UNION`.
10. Write an SQL query to find all information about the employees whose position is "SE" (Software Engineer).

Exercises (3)

11. Write an SQL query to find the names of all employees whose first name starts with "S". Use `LIKE`.
12. Write an SQL query to find the names of all employees whose last name contains the character sequence "an". Use `LIKE`.
13. Write an SQL query to find the names of all employees whose salary is in the range [3000...5000]. Use `BETWEEN`.
15. Write an SQL query to find the names of all employees whose salary is 2500, 4000 or 5000. Use `IN`.
16. Write an SQL query to find all employees that have no manager. Use `IS NULL`.