Welcome to the

Java Course

Module 4 – Day 02

Content of the course

- Introduction to Database Theory and SQL Basics
- Database connection with Java
- **Table Management and Relationships**
- Advanced SQL Queries and Integration with Java
- Data Normalization

Database Overview

- What are Databases?
- Why use Databases?

- Databases are systems that allow users to store and organize data.
- They are useful when dealing with large amounts of data.

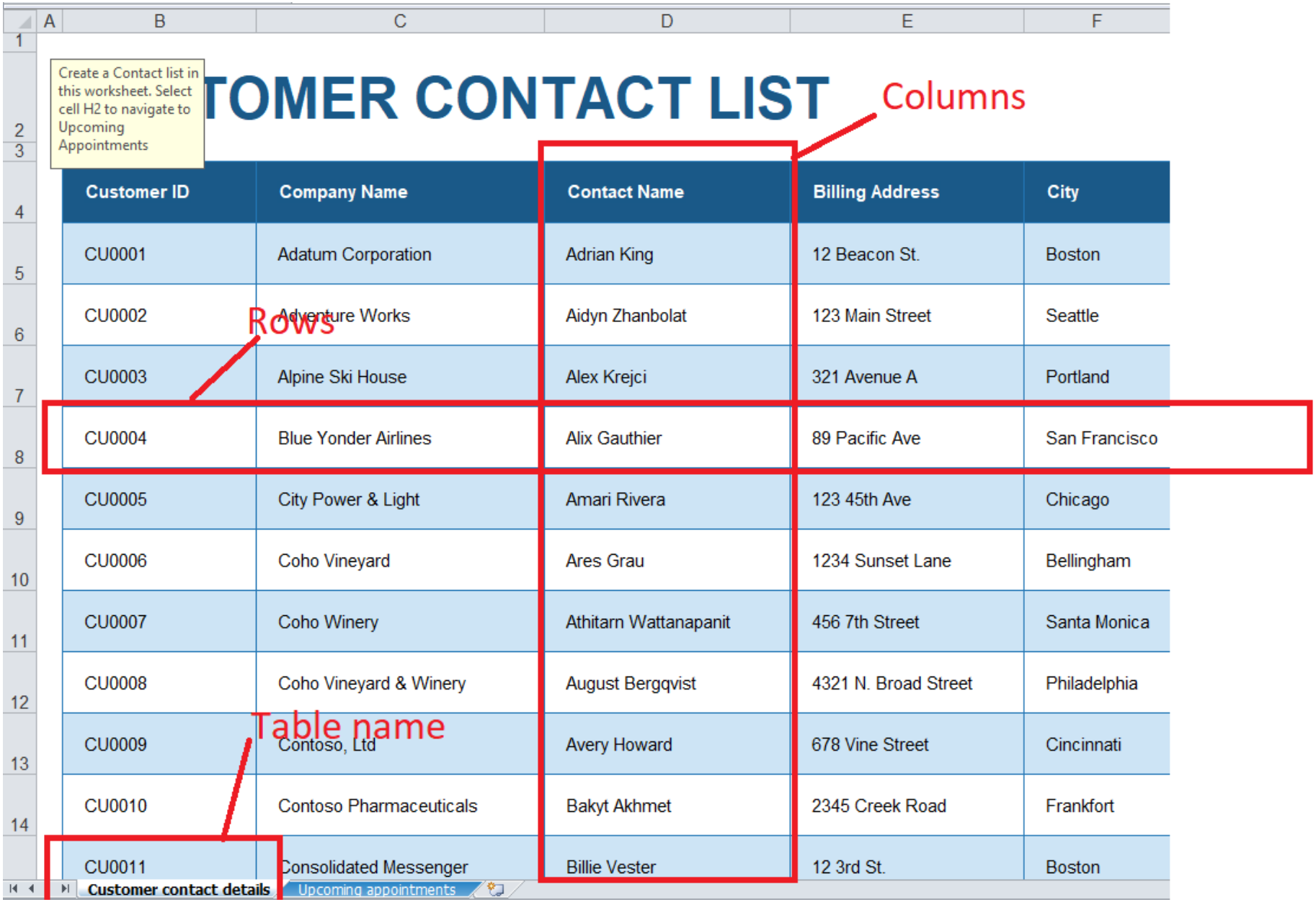
Typical users of Database

Analysts

- Marketing
- Business
- Sales

Technical

- Data Scientists
- Software Engineers
- Web Developers


Create a Contact list in this worksheet. Select cell H2 to navigate to Upcoming Appointments
TOMER CONTACT LIST
Columns
Rows
Table name
Customer ID
Company Name
Contact Name
Billing Address
City
CU0001
Adatum Corporation
Adrian King
12 Beacon St.
Boston
CU0002
Adventure Works
Aidyn Zhanbolat
123 Main Street
Seattle
CU0003
Alpine Ski House
Alex Krejci
321 Avenue A
Portland
CU0004
Blue Yonder Airlines
Alix Gauthier
89 Pacific Ave
San Francisco
CU0005
City Power & Light
Amari Rivera
123 45th Ave
Chicago
CU0006
Coho Vineyard
Ares Grau
1234 Sunset Lane
Bellingham
CU0007
Coho Winery
Athitarn Wattanapanit
456 7th Street
Santa Monica
CU0008
Coho Vineyard & Winery
August Bergqvist
4321 N. Broad Street
Philadelphia
CU0009
Contoso, Ltd
Avery Howard
678 Vine Street
Cincinnati
CU0010
Contoso Pharmaceuticals
Bakyt Akhmet
2345 Creek Road
Frankfort
CU0011
Consolidated Messenger
Billie Vester
12 3rd St.
Boston
Customer contact details
Upcoming appointments


ORACLE
SYBASE
IBM
DB2
Open Source
Widely Used
Multi Platform
Open Source
Widely Used
Multi Platform
MySQL
PostgreSQL
SQLite
ODBC
SAP HANA
Amazon SimpleDB

SQL Cheat Sheet

SQL SELECT STATEMENTS

SELECT * FROM tbl
Select all rows and columns from table tbl

SELECT c1,c2 FROM tbl
Select column c1, c2 and all rows from table tbl

SELECT c1,c2 FROM tbl
WHERE conditions
ORDER BY c1 ASC, c2 DESC
Select columns c1, c2 with where conditions and from table tbl order result by column c1 in ascending order and c2 in descending order

SELECT DISTINCT c1, c2
FROM tbl
Select distinct rows by columns c1 and c2 from table tbl.

SELECT c1, aggregate(expr)
FROM tbl
GROUP BY c1
Select column c1 and use aggregate function on expression expr, group columns by column c1.

SELECT c1, aggregate(expr) AS c2
FROM tbl
GROUP BY c1
HAVING c2 > v
Select column c1 and c2 as column alias of the result of aggregate function on expr. Filter group of records with c2 greater than value v

SQL UPDATE TABLE

INSERT INTO tbl(c1,c2,...)
VALUES(v1,v2...)
Insert data into table tbl

INSERT INTO tbl(c1,c2,...)
SELECT c1,c2.. FROM tbl2
WHERE conditions
Insert data from tbl2 into tbl

UPDATE t
SET c1 = v1, c2 = v2...
WHERE conditions
Update data in table tbl

DELETE FROM tbl
WHERE conditions
Delete records from table tbl based on WHERE conditions.

TRUNCATE TABLE tbl
Drop table tbl and re-create it, all data is lost

SQL TABLE STATEMENTS

```
CREATE TABLE tbl(
        c1 datatype(length)
        c2 datatype(length)
        ...
        PRIMARY KEY(c1)
)
```
Create table tbl with primary key is c1

DROP TABLE tbl
Remove table tbl from database.

ALTER TABLE tbl
ADD COLUMN c1 datatype(length)
Add column c1 to table tbl

ALTER TABLE tbl
DROP COLUMN c1
Drop column c1 from table tbl

SQL JOIN STATEMENTS

SELECT * FROM tbl1
INNER JOIN tbl2 ON join-conditions
Inner join table tbl1 with tbl2 based on join-conditions.

SELECT * FROM tbl1
LEFT JOIN tbl2 ON join-conditions
Left join table tbl1 with tbl2 based on join-conditions.

SELECT * FROM tbl1
RIGHT JOIN tbl2 ON join-conditions
Right join table tbl1 with tbl2 based on join-conditions.

SELECT * FROM tbl1
RIGHT JOIN tbl2 ON join-conditions
Full outer join table tbl1 with tbl2 based on join-conditions.

Section Overview

- Data Types
- Primary and Foreign Keys
- Constraints
- CREATE
- INSERT
- UPDATE
- DELETE, DROP, ALTER

Data Type

Boolean
- True or False

Character
- Char, varchar, text

Numeric
- Integer, floating point number

Temporal
- Date, time, timestamp and interval

https://www.postgresql.org/docs/current/datatype.html

Primary Key :

- A primary key is a column or a set of columns that uniquely identifies each record/row in a table.
- It must contain unique values for each row, and has NOT NULL values.
- There can be only one primary key in a table.
- Primary keys are typically used as the basis for relationships with other tables.

Foreign Key :

- A foreign key is a column or a set of columns in one table that refers to the primary key in another table.
- It establishes a link between two tables, enforcing ***referential integrity***.
- The foreign key constraint ensures that the values in the foreign key column(s) match values in the primary key column(s) of the referenced table or are NULL.
- It helps maintain consistency and integrity in the database by preventing actions that would cause orphaned records or invalid references.

Most Common Column Constraints used:

NOT NULL constraint
- Ensures that a column cannot have NULL value.

UNIQUE constraint
- Ensures that all values in a column are different.

PRIMARY Key
- Uniquely identifies each row/record in a database.

FOREIGN Key
- Constraints data based on columns in other tables.

CHECK constraint
- Ensures that all values in a column satisfy certain conditions. (ex: >18)

CREATE TABLE statement

CREATE TABLE statement allows you to define the structure of a new table by specifying the names and data types of its columns.

```
CREATE TABLE students (
    student_id INT PRIMARY KEY,
    name VARCHAR(50),
    age INT,
    email VARCHAR(100)
);
```

CREATE TABLE with References

Use **REFERENCES** keyword when creating a Table to reference 2 tables together

```
-- Create a third table 'enrollments' to represent the many-to-many relationship
CREATE TABLE enrollments (
    enrollment_id SERIAL PRIMARY KEY,
    student_id INT,
    course_id INT,
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (course_id) REFERENCES courses(course_id)
);
```

If you want to specify a constrain for a field:

`NOT NULL`	Cannot have `NULL` value
`UNIQUE`	Each value is unique
`PRIMARY KEY`	A combination of `NOT NULL` and `UNIQUE`
`FOREIGN KEY`	Prevents destruction of links
`CHECK`	Satisfy a condition
`DEFAULT`	If a value is not specifies, the default value will be given
`SERIAL` (only in PostgresQL)	Generates a sequence of integers

Now YOUR TURN !

Let’s do exercises 1.1, 1.2, 1.3

Insert data

INSERT INTO <table name> (col1, col2, …)
VALUES (val1, val2, …);

Example :

```
INSERT INTO account_test(user_name, password, email,
created_on)
VALUES
('Jose', 'password', 'jose@gmail.com',
CURRENT_TIMESTAMP);
```

Now YOUR TURN !

Let’s do exercises 1.4

Update data

```
UPDATE <table name>
SET col1 = val1, col2 = val2, …
WHERE <condition>;
```

Example :

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

Modifying tables

The **ALTER TABLE** statement allows you to modify the structure of an existing table.

```
ALTER TABLE students ADD COLUMN major VARCHAR(50);
```

Now YOUR TURN !

Let’s do exercises 1.5, 1.6

Delete data

DELETE FROM <table name>
WHERE <condition>;

Example :

DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste';

Deleting tables

The **DROP TABLE** statement allows you to delete an entire table from the database, along with all its data and structure.

```
DROP TABLE students;
```

Truncating table

- If you want to keep the structure of the table (columns, references), you can use **TRUNCATE TABLE**
- It will delete all the rows inside the table

```
TRUNCATE TABLE table_name;
```