Design Laboratory (CS69202)
Spring Semester 2025

Topic : MYSQL + NoSQL- Mapper, Combiner, Reducer
Date: January 22, 2025

Important Instructions:

1. There are 2 parts of this assignment [SQL & No-SQL]. For part 1, you need to do Sections A and B in the lab and Section C can be done at home for your practice. For part 2, Section A needs to be in the lab and Section B is for practice at your home.
2. For Part 1, you need to create the tables, insert the data and write the queries.
3. For Part 2 you need to write Mapper, Reducer and Combiner codes for the No-SQL Queries. You also need to write a Bash script which on executing gives the necessary outputs. Strictly follow this format else ZERO marks will be allocated.
4. The mapper, Combiner, and Reducer need to work as entitled if a code is found to work but WITHOUT correct mapper or reducer logic, then ZERO marks will be allotted.
5. You must NOT use any libraries apart from sys, datetime, or time module.
6. Data is present in the "data/" directory of each part.
7. Submit a single zip **<roll_number>_sql_nosql**. It will contain two folders "***SQL***" and *"**NO-SQL**"*. The "SQL" folder will contain all the queries in a <rollno>_queries.sql file and a document *<rollno>_output.pdf* that contains the screenshot of the query along with the output. The "NO-SQL" folder would contain all the *files for mapper, reducer, combiner, bashscript, input and output.*

PART -1 SQL

Database Schema Summary **[*order_supply_schema*]**

- Suppliers
    - Purpose: Stores supplier details.
    - Columns:
        - **Id** (PK): Unique supplier ID. [int]
        - **CompanyName**: Supplier company name. [varchar]
        - **ContactName**: Supplier contact person. [varchar]
        - **ContactTitle**: Supplier contact title [varchar]
        - **City:** Supplier location city [varchar]
        - **Country**: Supplier location country [varchar]
        - **Phone**: Contact details. [varchar]

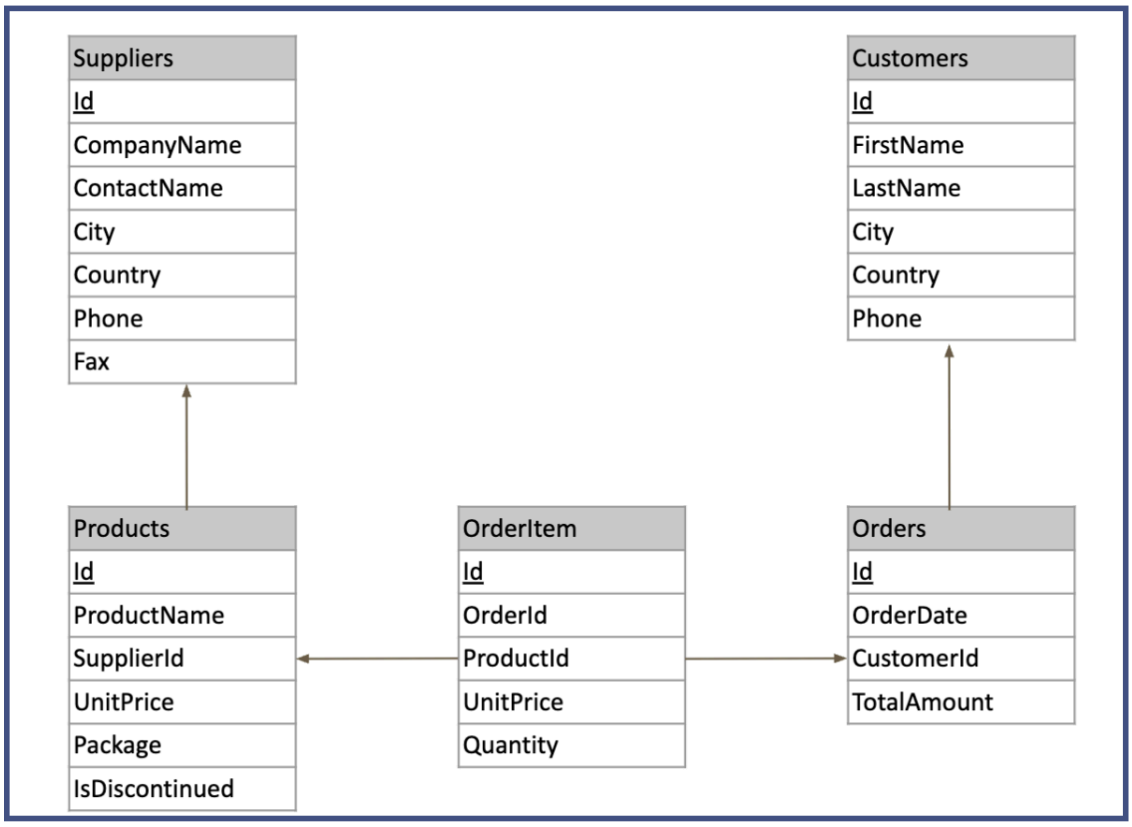
        - **Fax**: Contact Fax details [varchar]

- Products
    - Purpose: Store product details.
    - Columns:
        - **Id (PK)**: Unique product ID. [int]
        - **ProductName**: Name of the product. [varchar]
        - **SupplierId**: Linked supplier. [int]
        - **UnitPrice**: Price per unit. [decimal]
        - **Package**: Packaging type. [varchar]
        - **IsDiscontinued**: Product status (yes/no). [bit]

- OrderItem
    - Purpose: Details of order items.
    - Columns:
        - **Id (PK)**: Unique order item ID. [int]
        - **OrderId**: Linked order. [int]
        - **ProductId**: Linked product. [int]
        - **Quantity**: Ordered quantity. [decimal]
        - **UnitPrice**: Price per unit at the time of order. [int]

- Orders
    - Purpose: Stores order information.
    - Columns:
        - **Id (PK)**: Unique order ID. [int]
        - **OrderDate**: Order placement date. [datetime]
        - **OrderNumber** : Order number [varchar]
        - **CustomerId**: Linked customer. [int]
        - **TotalAmount**: Total order cost. [decimal]

- Customers
    - Purpose: Store customer details.
    - Columns:
        - **Id (PK)**: Unique customer ID. [int]
        - **FirstName**: Customer first name. [varchar]
        - **LastName**: Customer last name [varchar]
        - **City**: Customer city [varchar]
        - **Country**: Customer country. [varchar]
        - **Phone**: Contact number. [varchar]



Section 1 (INSERTING DATA) [5 + 10]

1. Identify foreign keys that link the tables
2. Create the order_supply_schema database with tables (*Suppliers, Products, OrderItem, Orders, Customers*), define primary and foreign key constraints, and insert data from the provided .sql file provided to you.

Section 2 (QUERIES SET -1) [3 + 4 + 4 + 4 + 5 + 5 + 5 +5]

1. Display the names of suppliers located in France.
2. Count the orders placed in January month of any year.
3. Display the names of customers who have placed at least three orders
4. Display the product name and unit price for products with a unit price greater than $100
5. Find the most popular product (the one with the highest total quantity ordered).
6. List the top 3 most expensive products
7. Identify customers who have not placed any orders in the year 2014.
8. Retrieve the details of discontinued products (name, unit price, supplier).

Section 3 (QUERIES SET -2)

9. Display the names of suppliers that provide more than 5 products.
10. Find the total revenue generated from orders placed by customers located in the United States, grouped by city.
11. Identify the products that customers have ordered from more than 3 different cities.
12. Calculate each supplier's average order value (average total number of orders for the products they supply).
13. Identify products that multiple suppliers have supplied.
14. Find the suppliers who have products with a unit price higher than the average unit price for all products in the database.
15. List the customers who have placed orders for products from suppliers located in more than 2 different countries.

PART -2 No-SQL

Section A: (In Class) [10 + 5 + 5 + 5 + 10 + 5 + 5 + 5]

Write Mapper, Combiner and Reducer codes for the following:

Q1. Your task is to generate the frequency distribution of the words across the entire dataset (i.e taking words from all the files).
Write bash script [10 Points] to run the mapper [5 Points], combiner [5 Points] and reducer [5 Points] on the files and store the result in *result_1.txt.*

Q2. Store the Unique words that have a frequency count of <10 across all the files.
Write bash script [10 Points] to run the mapper [5 Points], combiner [5 Points] and reducer [5 Points] on the files and store the result in *result_2.txt.*

[Approach : Use the combiner effectively to decrease the amount of input data to the reducer]

Example:

```
# mapper.py
import sys
filename = sys.argv[1]
with open(filename) as f:
        for i in f:
                i = i.replace("\n","")
                x = i.split()
                print(f"{x[0]}\t{1}")
```

```
# Combiner.py
import sys

current_word = None
current_count = 0

# Process each line of input
for line in sys.stdin:
    word, count = line.strip().split('\t')
    try:
        count = int(count)
    except ValueError:
        continue

    # Check if the current word matches the last seen word
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # Emit the word and its aggregated count
            print(f'{current_word}\t{current_count}')
        current_word = word
        current_count = count

# Emit the last word and its count
if current_word:
    print(f'{current_word}\t{current_count}')
```

```python
# reducer.py

import sys

current_word = None
current_count = 0

# Process each line of input
for line in sys.stdin:
    word, count = line.strip().split('\t')
    try:
        count = int(count)
    except ValueError:
        continue

    # Check if the current word matches the last seen word
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # Emit the word and its final count
            print(f'{current_word}\t{current_count}')
        current_word = word
        current_count = count

# Emit the last word and its count
if current_word:
    print(f'{current_word}\t{current_count}')
```

```bash
#!/bin/bash

# Directory containing your data files
DATA_DIR="./data"

# Temporary file to store intermediate results
INTERMEDIATE_FILE="intermediate.txt"

# Output file for the final result
OUTPUT_FILE="result__.txt"

# Ensure the intermediate file is empty
> "$INTERMEDIATE_FILE"

# Loop through each text file in the data directory
for file in "$DATA_DIR"/*.txt; do
    # Process each file with the mapper, sort the output, then pass it to the combiner
    echo "Processing file: $file"
    python mapper.py "$file" | sort | python combiner.py >> "$INTERMEDIATE_FILE" &
done

# Wait for all background processes to complete
wait

# Sort the combined intermediate results and pass them to the reducer
sort -n "$INTERMEDIATE_FILE" | python reducer.py > "$OUTPUT_FILE"

# Clean up the intermediate file
rm "$INTERMEDIATE_FILE"

```

Section B (Take Home)

Q1. Write python code to split the given file into 100 files. Use these files as the input for the following questions.
Write Mapper, Combiner and Reducer codes for the following.
Also write the bash file to generate the outputs, and save them as *results_B2.txt*, r*esults_B3.txt*, etc.

Q2. Determine the total number of medals won by each country across all Olympic events

Q3. Identifying the Most Successful Athlete in Each Discipline

Q4. Determine the Number of Unique Athletes per Country

Q5. Identifying the Most Common Event Type per Year