Answers to week-1 – assignment (datababe)

1. **Part 1: <u>Understanding SQL</u>**

   1.1 **SQL** (Structured Query Language) plays an important role in managing data for dynamic websites, including online store. SQL to unlock the power to store, manage, delete and retrieve data in the web application. This is how SQL work behind the scenes: SQL query - Query language processor - DBMS engine -physical database

1.2 ROLE OF SQL IN WEB APPLICATIONS

In web applications, **SQL (Structured Query Language)** plays a pivotal role in managing data behind the scenes. Here's how it contributes:

1. **Data Storage**: SQL databases store critical information such as user profiles, product details, and transaction records. These databases ensure data consistency and integrity.
2. **Query Processing**: Developers use SQL queries to retrieve, update, and manipulate data. For instance, when a user searches for products or places an order, SQL queries handle these interactions.
3. **Relational Structure**: SQL databases organize data into tables with relationships (e.g., one-to-many, many-to-many). This structure enables efficient data retrieval and maintains referential integrity.

1.3 THREE BENEFIT OF USING SQL FOR WEB APPLICTIONS

- Data Integrity
- Efficient Queries
- Scalability

1.4 EFFICIENCY, DATA RETRIEVAL CAPABILITIES, DATA ORGANIZATION

1. **Efficiency**: SQL databases optimize data storage, retrieval, and processing. They use indexing, caching, and query optimization techniques to minimize resource usage and response time.
2. **Data Organization**: SQL enforces a structured format for data storage, ensuring consistency and reducing redundancy. Tables, relationships, and constraints organize data logically.

3. **Data Retrieval**: SQL queries allow precise data extraction based on conditions. Developers can retrieve specific records, aggregate data, and join related tables efficiently.

1.5 LIST ANY THREE DATABASE MNAGAEMENT SYSTEM

- MySQL
- MariaDB
- Microsoft SQL Server

Part 2: Database Fundamentals

2.1 TABLES

→ **Database Table**: A database table is a fundamental structure within a relational database. It consists of rows (also known as records or tuples) and columns (also known as fields or attributes). Each row represents a specific instance of data, while each column defines a specific type of information (e.g., name, age, price). Tables are used to organize and store related data in a structured manner.

→ **Similarity to a Spreadsheet**: A database table is similar to a spreadsheet in that both use a grid-like format with rows and columns. In a spreadsheet, each cell corresponds to a specific data point, and formulas can be applied to manipulate data. Similarly, in a database table, rows contain individual records, and columns define the attributes of those records. However, databases offer additional features such as data integrity constraints, indexing, and efficient querying, making them more powerful for managing large-scale data compared to spreadsheets.

2.2 COLUMNS

A. **Columns**:

→ In a database table, **columns** represent individual data attributes or fields. Each column corresponds to a specific type of information (e.g., name, age, price).

For example, consider an **"Employee"** table with columns like **"EmployeeID"**, **"FirstName"**, **"LastName"**, and **"Salary"**. Each column stores a different type of data related to employees.

B. **Importance of Data Types**:

→ **Data Integrity**: Data types ensure that values stored in columns match their intended format (e.g., text, numeric, date). This prevents inconsistencies and errors.

→ **Storage Efficiency**: Choosing appropriate data types optimizes storage space. For instance, using an integer type for employee IDs is more efficient than using text.

→ **Query Performance**: Data types of impact query execution speed. Efficiently indexed and typed columns enhance search and retrieval operations.

2.3 Common Data Types:

→ **Text (VARCHAR or CHAR)**: Stores character strings (e.g., names, descriptions). Variable-length (VARCHAR) or fixed-length (CHAR) options are available.

→ **Number (INT, FLOAT, DECIMAL)**: Represents numeric values (e.g., quantities, prices). Integer (INT), floating-point (FLOAT), and decimal (DECIMAL) types are common.

→ **Date (DATE or DATETIME)**: Stores dates and times (e.g., hire dates, order timestamps). DATE represents a date, while DATETIME includes both date and time components.

2.3 DATA TYPES

Data types play a crucial role in ensuring data integrity and efficient storage within databases. Here's why they matter:

1. **Data Integrity**: By defining data types, we enforce rules for valid values (e.g., text, numbers, dates). This prevents incorrect or incompatible data from being stored, maintaining consistency and reliability.
2. **Storage Efficiency**: Choosing appropriate data types optimizes space usage. For instance:
    - **Text (VARCHAR)**: Variable-length character strings (e.g., names, descriptions) conserve storage by allocating only what's needed.
    - **Number (INT)**: Integer data type efficiently stores whole numbers (e.g., employee IDs) without decimal precision.

- **Date (DATE)**: Stores calendar dates (e.g., hire dates) compactly, using fewer bytes than text representations.

PART3: EXPENSE TRACKER DATABASE DESIGN

3.1 PLANNING: DATA TYPE REQUIRED TO BUILD AND EXPENSE TRACKER APPLICATION

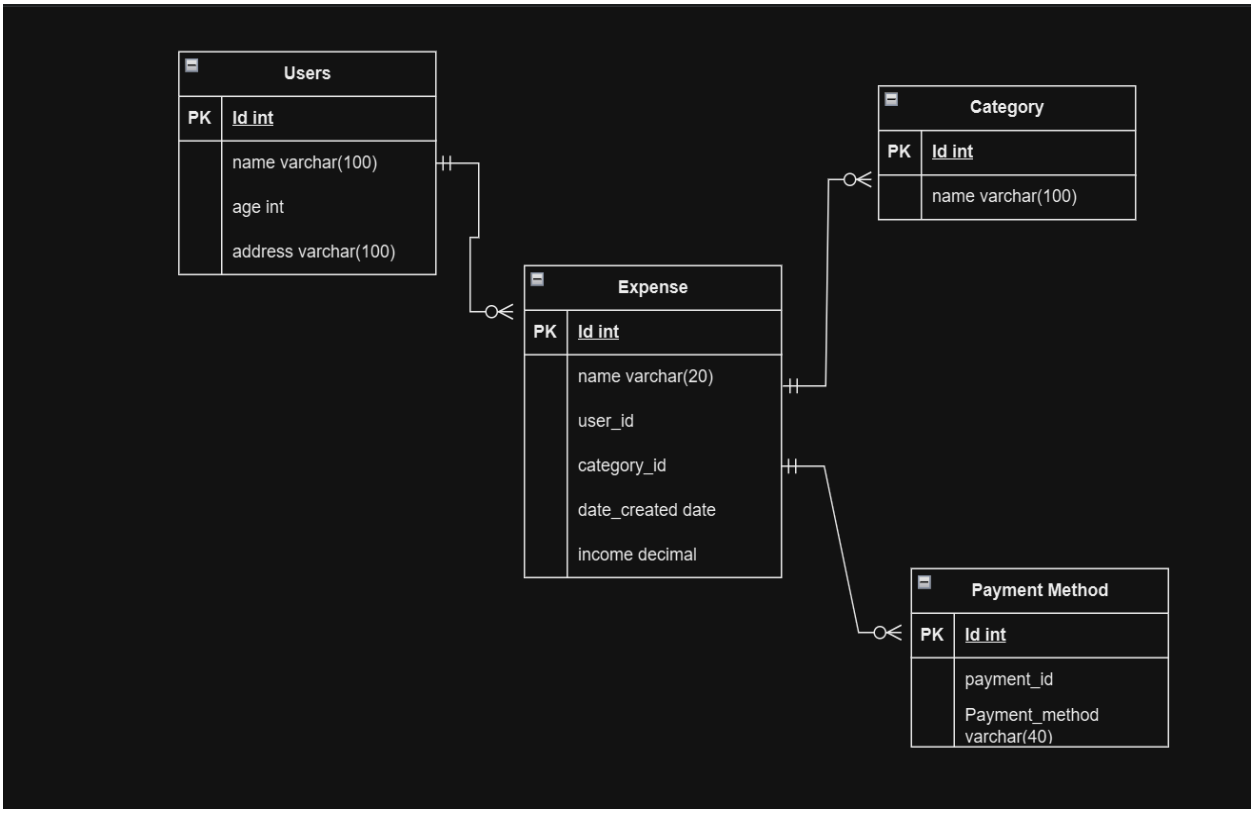
When building an **Expense Tracker application**, you'll need to track several data points related to expenses. Here are **five relevant data points**:

- **Expense Amount**: The cost of the expense (e.g., $50.00 for groceries).
- **Date**: The date when the expense occurred (e.g., June 15, 2024).
- **Category**: The category or type of expense (e.g., food, transportation, utilities).
- **Payment Method**: How the expense was paid (e.g., cash, credit card, online transfer).
- **Description**: Additional details about the expense (e.g., "Lunch with colleagues").

3.2 TABLES: BASIC DATABASE SCHEMA WITH ONE MAIN TABLE

Let's design a basic database schema for the **Expenses** table. We'll define the necessary columns and assign appropriate data types:

1. **Table Name**: Expenses
2. **Column Names**:
   - **Expense ID**: A unique identifier for each expense (e.g., auto-incremented integer).
   - **amount**: The expense amount (e.g., $50.00) stored as a decimal.
   - **date**: The date when the expense occurred (e.g., June 15, 2024) stored as a date.
   - **category**: The type of expense (e.g., food, transportation) stored as text.
   - **payment_method**: How the expense was paid (e.g., cash, credit card) stored as text.
3. **Data Types**:
   - **Expense_ID**: INT (auto-incremented)
   - **amount**: DECIMAL
   - **date**: DATE
   - **category**: VARCHAR (text)
   - **payment_method**: VARCHAR (text)


Users
PK
Id int
name varchar(100)
age int
address varchar(100)
Category
PK
Id int
name varchar(100)
Expense
PK
Id int
name varchar(20)
user_id
category_id
date_created date
income decimal
Payment Method
PK
Id int
payment_id
Payment_method varchar(40)