WORKSHEET 1 SQL

1. **Which of the following is/are DDL commands in SQL?**

Answer :- C & D

2. **Which of the following is/are DML commands in SQL?**

Answer :- A & B

3. **Full form of SQL is:**

Answer :- B

4. **Full form of DDL is:**

Answer :- B

5. **DML is:**

Answer :- A

6. **Which of the following statements can be used to create a table with column B int type and C floattype?**

Answer :- C

7. **Which of the following statements can be used to add a column D (float type) to the table A created above?**

Answer :- B

8. **Which of the following statements can be used to drop the column added in the above question?**

Answer :- B

9. **Which of the following statements can be used to change the data type (from float to int) of the column Dof table A created in above questions?**

Answer :- B

10. **Suppose we want to make Column B of Table A as primary key of the table. By which of the following statements we can do it?**

Answer :- C

11. What is data-warehouse?

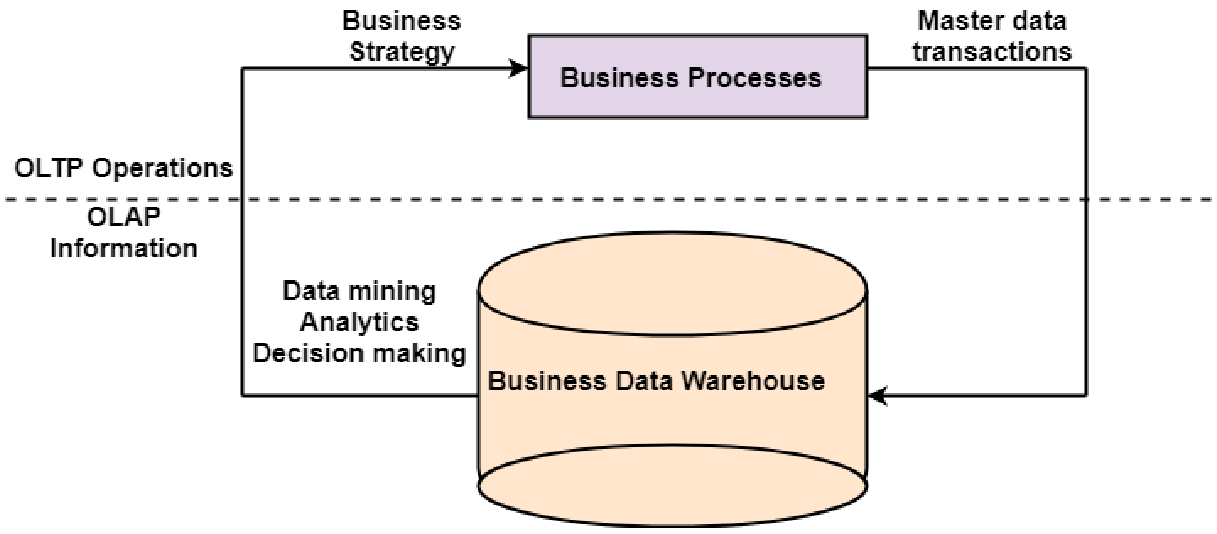
Answer :- A data warehouse is a central repository of information that can be analyzed to make more informed decisions. Data flows into a data warehouse from transactional systems, relational databases, and other sources, typically on a regular cadence. Business analysts, data engineers, data scientists, and decision makers access the data through business intelligence (BI) tools, SQL clients, and other analytics applications.

Data and analytics have become indispensable to businesses to stay competitive. Business users rely on reports, dashboards, and analytics tools to extract insights from their data, monitor business performance, and support decision making. Data warehouses power these reports, dashboards, and analytics tools by storing data efficiently to minimize the input and output (I/O) of data and deliver query results quickly to hundreds and thousands of users concurrently.

12. What is the difference between OLTP VS OLAP?

Answer :- **OLTP (On-Line Transaction Processing)** is featured by a large number of short on-line transactions (INSERT, UPDATE, and DELETE). The primary significance of OLTP operations is put on very rapid query processing, maintaining record integrity in multi-access environments, and effectiveness consistent by the number of transactions per second. In the OLTP database, there is an accurate and current record, and schema used to save transactional database is the entity model (usually 3NF).

OLAP (On-line Analytical Processing) is represented by a relatively low volume of transactions. Queries are very difficult and involve aggregations. For OLAP operations, response time is an effectiveness measure. OLAP applications are generally used by Data Mining techniques. In OLAP database there is aggregated, historical information, stored in multi-dimensional schemas (generally star schema).



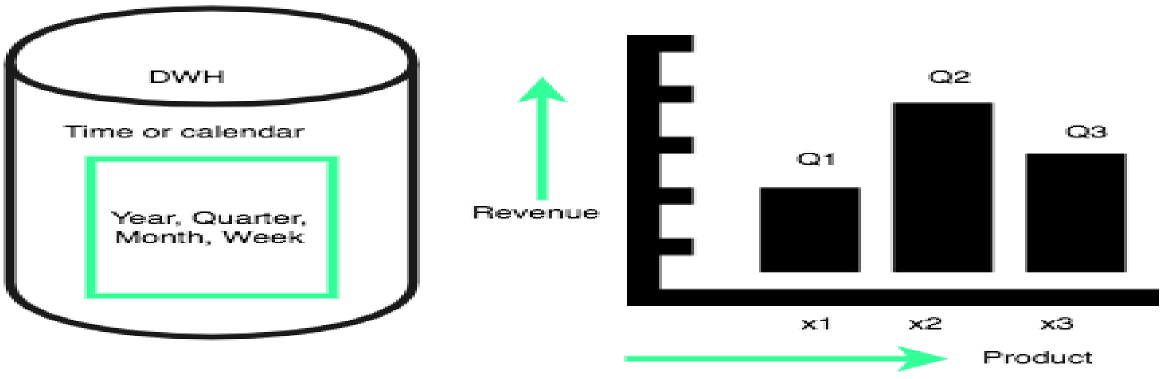
13. What are the various characteristics of data-warehouse?

Answer :- Data Warehouse is designed with four characteristics. They are

1. Time variant.
2. Non Volatile.
3. Integrated.
4. Subject Oriented.

Time Variant

A Data Warehouse is a time variant data base, which supports the business management in analysing the business and comparing the business with different time periods like Year, Quarter, Month, Week and Date.



Attributes of Time

- DAY_NAME
- DAY_NUMBER_IN_WEEK
- DAY_NUMBER_IN_MONTH
- DAY_NUMBER_IN_YEAR
- WEEK_NUMBER_IN_MONTH
- WEEK_NUMBER_IN_YEAR
- MONTH_NUMBER
- MONTH_YEAR
- QUARTER_YEAR
- QUARTER_NUMBER

- YEAR
- SESSION
- WEEKEND_INDICATOR_FLAG
- WEEKDAY_INDICATOR_FLAG

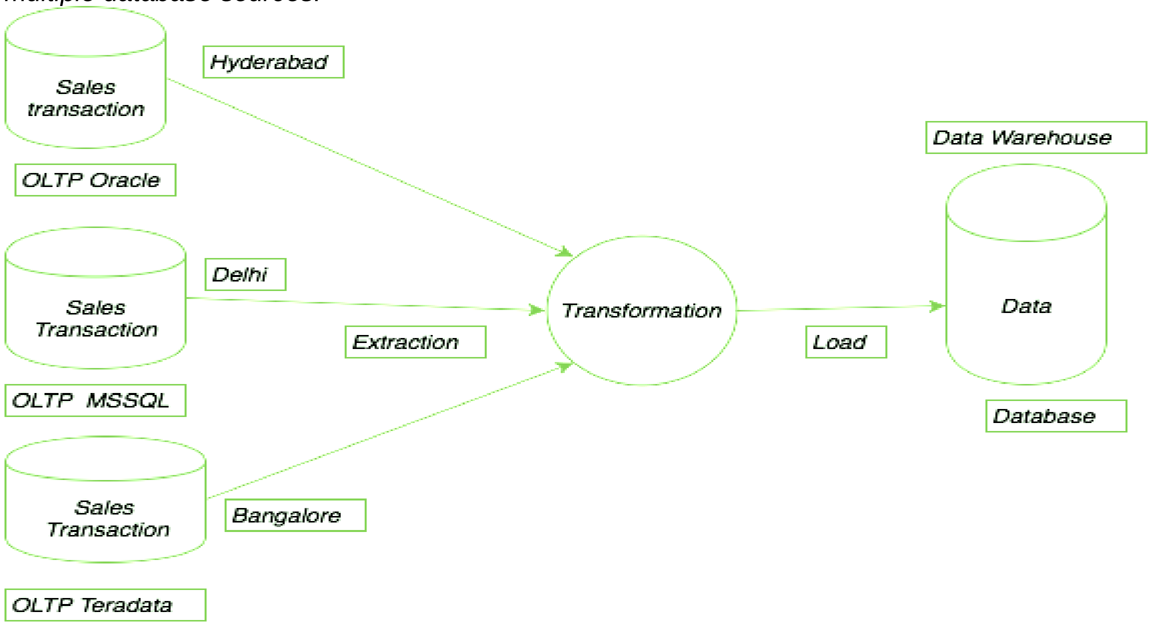
Non Volatile

It is non volatile Database, once the data entered into the database, it does not reflects to the change which takes place at operational database. Hence the data is statics in Data Warehouse.

- It generates artificially keys or surrogate keys to store the history.
- A surrogate key generated serious of numbers.
- It requires more disk space.

Integrated Database

A DWH is a integrated database, which allows you to collect the data and integrate the data with multiple database sources.

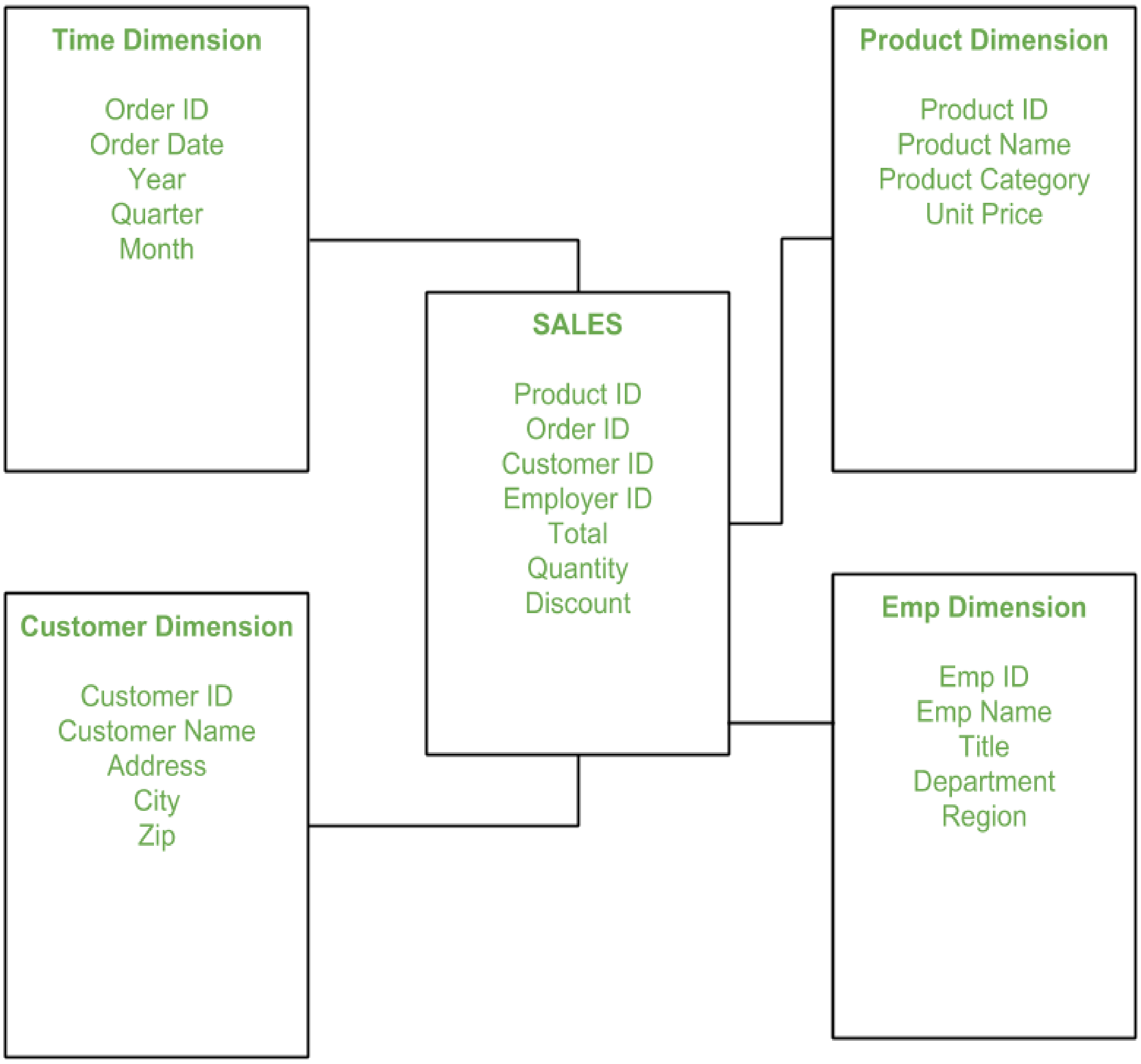


Subject Oriented

Data warehouse is a subject oriented database, which supports the business need of individual department specific user.
Example : Sales, HR, Accounts, Marketing etc.

14. What is Star-Schema??

Answer :- **Star schema** is the fundamental schema among the data mart schema and it is simplest. This schema is widely used to develop or build a data warehouse and dimensional data marts. It includes one or more fact tables indexing any number of dimensional tables. The star schema is a necessary cause of the snowflake schema. It is also efficient for handling basic queries.
It is said to be star as its physical model resembles to the star shape having a fact table at its center and the dimension tables at its peripheral representing the star's points. Below is an example to demonstrate the Star Schema:



In the above demonstration, SALES is a fact table having attributes i.e. (Product ID, Order ID, Customer ID, Employer ID, Total, Quantity, Discount) which references to the dimension tables. **Employee dimension table** contains the attributes: Emp ID, Emp Name, Title, Department and Region. *Product dimension table* contains the attributes: Product ID, Product Name, Product Category, Unit Price. *Customer dimension table* contains the attributes: Customer ID, Customer Name, Address, City, Zip. *Time dimension table* contains the attributes: Order ID, Order Date, Year, Quarter, Month.

15. What do you mean by SETL ?

Answer :- SETL is a very-high level language with dynamic typing and dynamic data structures, based on the mathematical notion of set. It was designed in the very early 1970s by J. Schwartz – a renown mathematician, with the help of R. Dewar and others. The language introduced a fundamentally new paradigm in programming in which sets, ordered sets and maps are the principal data structures and the programs are expressed in terms of set constructors, set operations, and predicates on sets. The very name SETL is an abbreviation of 'SET Language'.

The set-oriented paradigm is based on the assumption that sets are as essential constructions in programming as they are in mathematics. SETL not only has extensive provision for programming with sets but also takes advantage of the syntactic tradition of abstract mathematics, the language's notation being very close to the one of set theory, thus making it possible to express many algorithms in a familiar, natural, and concise manner.

SETL programs are much more declarative than procedural. According to its author, the language should present 'an abstract but nevertheless executable notation for describing algorithms'. In his view, SETL should provide means for solving programming problems more or less at the semantic level of the problems themselves, postponing possible choices of more detailed encoding 'until logical structure is worked out'. This implies that SETL programs would typically be considered prototypes rather than real implementations. On the other hand, the implementations should be obtainable by staged refinement, also aided by the language.

The accent on prototyping came from the apprehension that a highly expressive language is necessarily inefficient. In fact, the (in)efficiency factor was probably exaggerated even then, and is certainly not that important now, with the mature compilation techniques and the much faster computers of today.

The primitive datatypes of SETL include integer, floating-point, Boolean, atom, and string. Integers are of unlimited magnitude. Atoms are unique values produced by a dedicated operation and are used to tag other values as an aid in constructing 'data structuring maps'. Strings, although counting as a primitive datatype, are operationally similar to ordered sets.

The name om for omega (Ω) represents an unknown or missing value. Like other values, it can be assigned to a variable, passed as an argument, returned as a functional result, and checked against, but cannot be used as an argument of an operator.

Both sets and ordered sets (called tuples) are heterogeneous in SETL: values of all sorts can be members of the same structure, including other sets and tuples. Nesting is unrestricted. Beyond sets and tuples, maps are the only other data structure provided in the language. A map is actually a set (and is denoted no differently) – one whose elements are pairs, i.e. tuples of length two. The sets of the first and the second items of those pairs constitute the domain and the range of the map, correspondingly. Although maps are not a separate datatype, they are supported in SETL by specific operations, besides those for sets.

The operations on sets available in SETL are the usual set-theoretic ones, among others intersection, union, difference, as well as tests for element and (sub)set inclusion. Selection of individual elements of a tuple or a map is done by subscripting. A slice denotation, e.g. t(i..j) refers to a range within a tuple or a string. Tuples of variables can appear on the left-hand side of an assignment statement to specify that assignings are simultaneous. For tuples whose content is an arithmetic progression there are special forms, namely [first..last] and [first,next..last]. Similar forms can be used for unordered sets, e.g. {2..5} is the same set as {4,3,2,5}.

Repetition is supported in SETL in several forms. One is a multi-part loop statement with optional initialization, pre- and post-conditions, steps at start and at end, and termination, as well as other facilities. More set-oriented is the kind of loop with set/tuple iterators of the form (variable in set) or (variable in set | condition). Finally, there are the so called compound operators, apparently borrowed from APL's reduce.

SETL's iterators are also used to construct sets and tuples through set/tuple formers. An example of a set former is

```
{2*x: x in s| x mod 5 = 0}
```

– the set of the doubled numbers found within the set or tuple s which are evenly divisible by 5. The language admits convenient simplifications of the general syntax of a former in obvious cases.

Along with comparisons and Boolean operators, for constructing Boolean expressions, e.g. within conditional or loop statements, SETL offers quantified tests. These have the general form ‘quantifier iterator | test’, where the quantifier is one of exists, notexists, and forall. When used in a conditional or a loop statement, a quantified test binds a name to a (the) specific value that satisfies the test, so that the actions in the respective construct could use that value. See the RPN calculator implementation in SETL for examples of quantified tests.

Apart from, and in addition to, the declarative style characteristic of SETL, the language has constructs that make it possible to use it in expression-oriented manner. Similar to Algol 68, C and elsewhere, assignments produce values and there are assigning operators: a sequence such as a := 3; b := 5; print(a *:= b +:= 2); would set b to 7, a to 21, and print 21. Along with conditional statements, there are expressions of that kind (if, case). And, in order to turn a statement sequence into an expression, one can make use of an expr...end compound statement: a yield statement within such a construct terminates it emitting a value.

For breaking a program down into callable pieces, SETL defines three kinds of entities: procedures, operators and refinements. A procedure can be made to return a value and so serve as a function. In fact, any procedure can be called as a statement or as a function, producing om in case it does not actually return a value. Unary and binary operators can be defined in order to enhance the expression-orientation of a program. Refinements are inlineable pieces of code that have names but no parameters. Although it is declared remotely, a refinement executes in the lexical environment of the point at which it is called.

The definitive book on SETL is ‘J. Schwartz et al. Programming with Sets: An Introduction to SETL, Springer, 1986’. Apart from it, there are not many reading sources on the language, and the available ones are incomplete. The best one available on-line is that of R. Dewar (see the links below).

For me, the contribution of SETL to the programming culture is not only the language itself but, at least as importantly, the thinking behind and about the language. In this respect, see the notes on the design of SETL linked at below, discussing such topics as: the advantages of having a highly expressive language for program prototyping (I wonder, was the very notion of prototyping articulated before SETL), declaratory approach to data and control structuring, application of operators in an object-dependent manner on a variety of data objects, avoiding repetition of detail, non-deterministic instruction sequencing, and footnoted style of expression and remote code dictions.

Although SETL is not, and has never been, very popular among practitioners, it must have had influence in the design of other languages. The set/list comprehensions in Miranda, Haskell and elsewhere spring to mind. With or without the influence of SETL, set-related constructs are not foreign to today's programming, as witnessed by Icon, the many languages that model sets through sequences of a kind, associative tables or other constructs, or by C++'s and Java's standard libraries, among other languages. It should be reiterated, however, that the set-oriented paradigm is more productive when backed up by a corresponding syntax, as opposed to merely calling library procedures.

SETL itself has evolved. One of the two most noticeable dialects is SETL2. It was designed as a development of SETL, adding named packages in the style of Ada, first-class procedures including anonymous ones, closures, and some built-in procedures for text processing like the ones in Snobol. Regretably, despite what its name may suggest, SETL2 is incompatible with SETL, omitting a number of features of the latter, and changing the syntax of others. For example, SETL2 does not have user-defined operators, the expr...end compound expression, and the very general multipart loop construct of SETL.

The other dialect to mention is ISETL, an interactive and more distant relative of SETL, known to be used, or to have been used, in teaching collegiate mathematics.

Recently (2012), an update of the language under the name SetlX appeared, seeking for renewing attention to SETL.