**ACID**: Atomicity, consistency, Isolation and durability

**Atomicity:** In a transaction involving two or more discrete pieces of information, either all of the pieces are committed or none are.

**Consistency:** A transaction either creates a new and valid state of data, or, if any failure occurs, returns all data to its state before the transaction was started.

**Isolation** A transaction in process and not yet committed must remain isolated from any other transaction.

**Durability:** Committed data is saved by the system such that, even in the event of a failure and system restart, the data is available in its correct state.

**Normalization** is a process of representing each entity with a relation. If you follow Entity Relationship Modeling (ERM), you don’t need normalization. You will apply normalization only to reinforce and ensure that the model is correct.

1NF The first normal form says that the tuples (rows) in the relation (table) must be unique, and attributes should be atomic.

**Schema:** A database contains schema and schema contains objects. Schema defines, how the objects must be represented.

**DataWarehouse** supports complete organization, data mart support a particular department. The simplest data warehouse design is called a star schema. The star schema includes several dimension tables (single table) and a fact table.

If you normalize the dimension, which results in multiple tables (snowflakes dimension). Schema contains snowflakes dimensions, then table dimensions is known as a snowflake schema

**Alter** command to modify the table. (Adding primary key, removing)

**SELECT**

**FROM**

**WHERE**

**GROUP BY**

**HAVING**

**ORDER BY;**

should follow that sequence. Group by should be after where, having after group by

ALTER TABLE Employee

ADD CONSTRAINT PK\_Name

PRIMARY KEY (Name)

-------

ALTER TABLE dbo.Employees

ADD CONSTRAINT CHK\_Employees\_salary

CHECK(salary > 0.00);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Delete a particular row or the rows with name = ‘rahul’**

delete from emp where name = 'rahu'

note: Delete will not work if their any reference(constraint) for that tuple in a different table.

Delete a particular column in a table

alter table emp

drop column age

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Here we list empid’s and how many customers they served in a year

SELECT empid, YEAR(orderdate) AS orderyear, COUNT(custid)

FROM Sales.Orders

GROUP BY empid, YEAR(orderdate)

But this will include duplicates. They might have served the same customer twice or more. Only to get the list of different customers. Use DISTINCT

SELECT empid, YEAR(orderdate) AS orderyear, COUNT(DISTINCT custid)

FROM Sales.Orders

GROUP BY empid, YEAR(orderdate)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The **HAVING** clause without a **GROUP BY** clause acts like the **WHERE** clause. If the **HAVING** clause contains no aggregate functions, use the **WHERE** clause for faster performance.

Having class is used like where class after applying group by

SELECT empid, YEAR(orderdate) AS orderyear

FROM Sales.Orders

WHERE custid = 71

GROUP BY empid, YEAR(orderdate)

HAVING COUNT(\*)>1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DISTINCT** key is used to remove duplicate rows. (only if the complete row is duplicate)

SELECT DISTINCT empid, YEAR(orderdate) AS orderyear

FROM Sales.Orders

WHERE custid = 71;

**Order by should always come in the last and the only clause to refer column alias**

SELECT TOP(5) PERSENT empid, YEAR(orderdate) AS orderyear, COUNT(\*) AS numorders

FROM Sales.Orders

WHERE custid = 71

GROUP BY empid, YEAR(orderdate)

HAVING COUNT(\*) > 1

ORDER BY empid ASC, orderyear DESC;