**DWH- Data Warehouse**

RDBMS or Database: In this the concerned person can read or write data and which can be changed after every update, insert or modification, which will be impacted.

If we want to convert RDBMS or database into DWH then it should have below features to be applicable.

**DWH:**

DWH is defined as Subject Oriented, Integrated, time variant and non-volatile collection of data for decision making.

**subject oriented**: This is used to analyses subject area.

**Integrated:** It should integrate data from different sources.

**Time Variant:** Historical data is usually maintained in a data warehouse, i.e. retrieval can be for any period.

**Non-Volatile:** Once data is stored in data warehouse, it cannot be altered.

**Data ware house concepts**

**OLTP** - Online Transaction Processing system

OLTP is Nothing but a database which stores daily transactions which is called current data.

OLTP is used more for online applications, where application needs to update very frequently to maintain consistency of data.

**OLAP** - Online analytical processing

OLAP delas with analyzing data for decision making and planning. It will maintain historical data.

**COB** - Close of business - 9AM to 4PM

**DATA MART:**

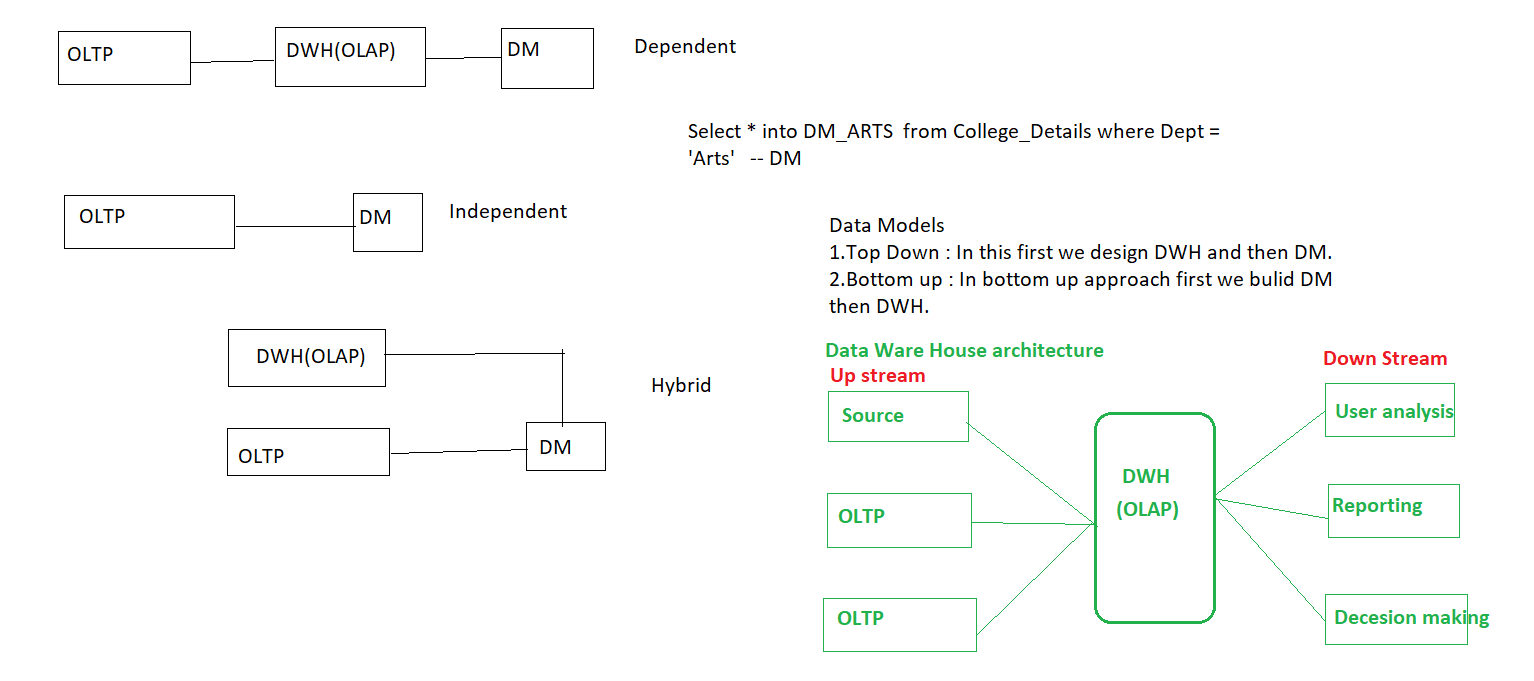
* It is smallest version of DWH.
* DATA MART deals with single subject area.
* DATA MART focuses on one area hence they carry data from limited data source.

**DATA MART types**

**1.Dependent:** Dependent DM are created from DWH.

**2.Independet:** Independent DM are created directly from source system and suitable for small organization

**3.Hybrid:** In Hybrid DM data can be feed from data warehouse or operational data (OLTP).



**Data Models**

* Data Model tells how the logical structure of database is modeled.
* Data Models are fundamental entities to introduce ideas in RDBMS.

***Data Models defines how data is connected to each other and how it will be processed and stored inside the system.***

* Data Models are designed by Principal Architect/ Data Base Solution Engineer and designed document is called as **HLSD** (High Level Design Specification Document) / **HLD** (High Level design)/**DSD** (Design Specification Document).

To design a final Data Models, they follow step – by -step approaches and those are

1.Conceptual

2. Logical

3. Physical

**For Example:**

For manufacturing unit need develop DATA MODEL then we have some below entities

PRODUCT

CUSTOMER

STORE

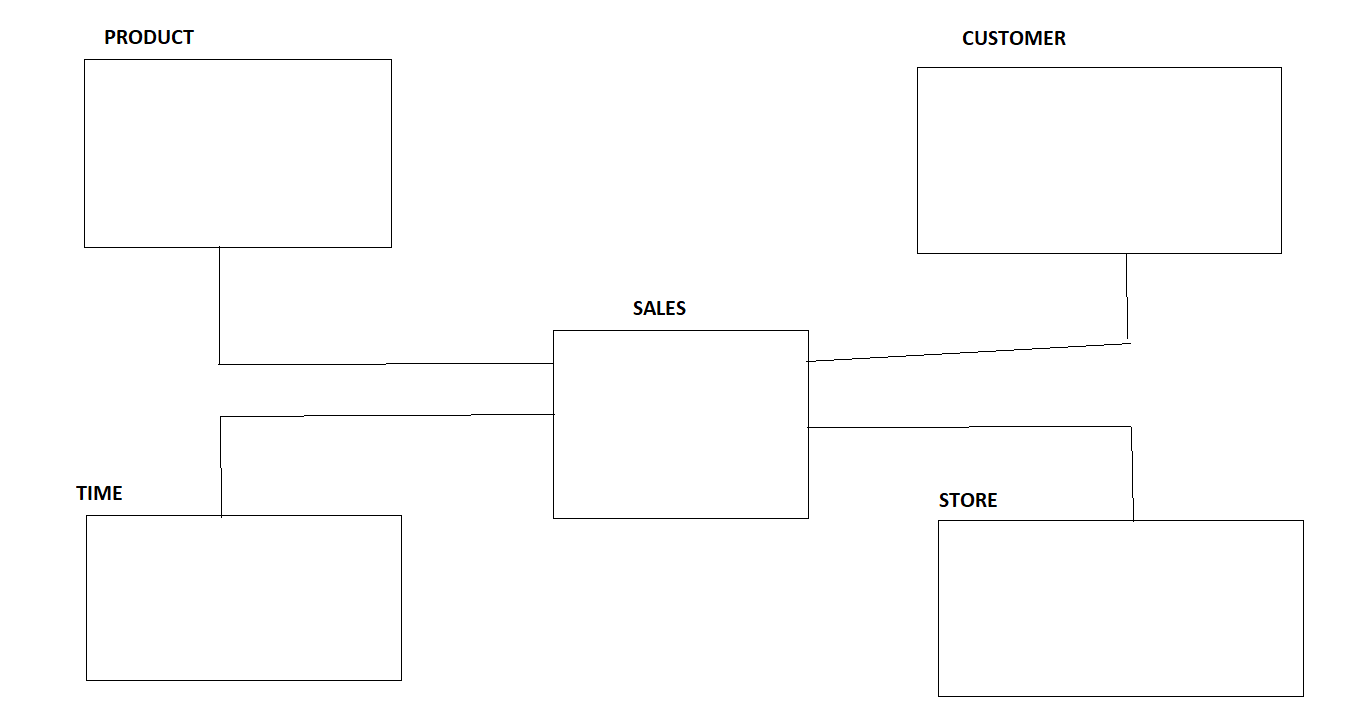
TIME

SALES

**1.Conceptual**

This model usually pictures highest level of relationship between entities

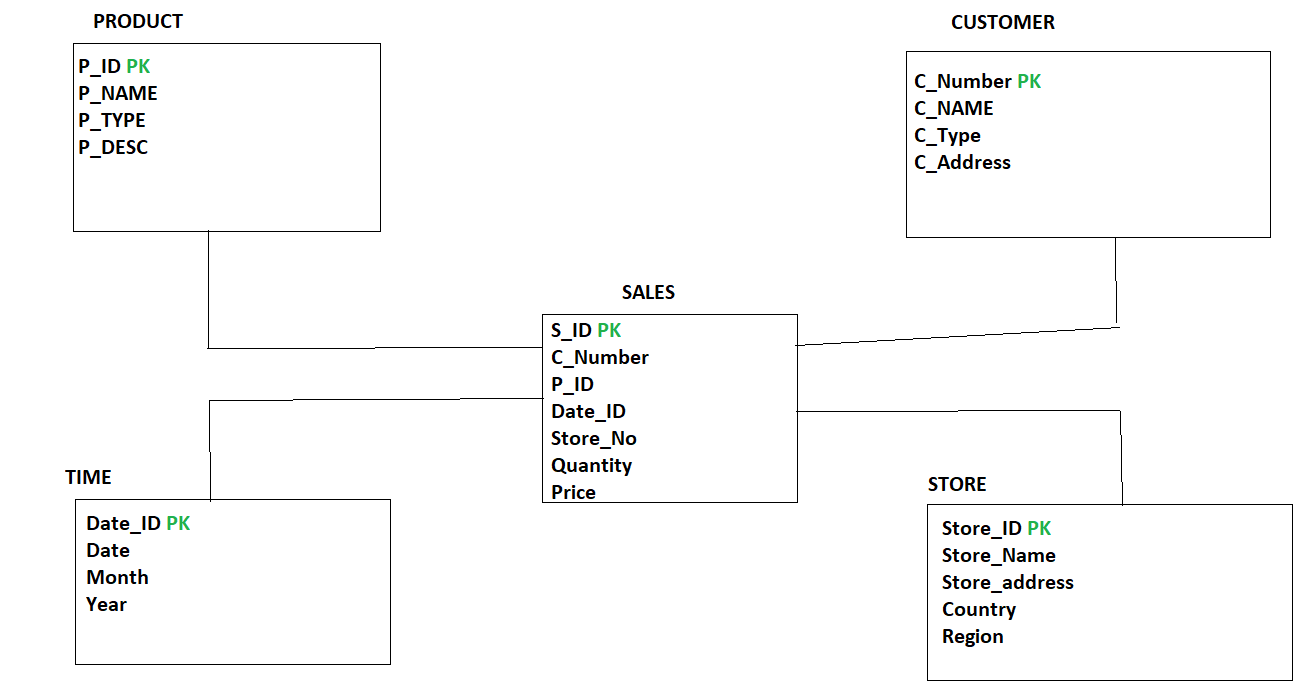
* Displays important entities and relationships among them.
* No attributes or columns specified.
* No primary key is associated or defined.



**2.Logical**

Logical Data Model defines data as much as possible, To show how they can be physically implemented in database.

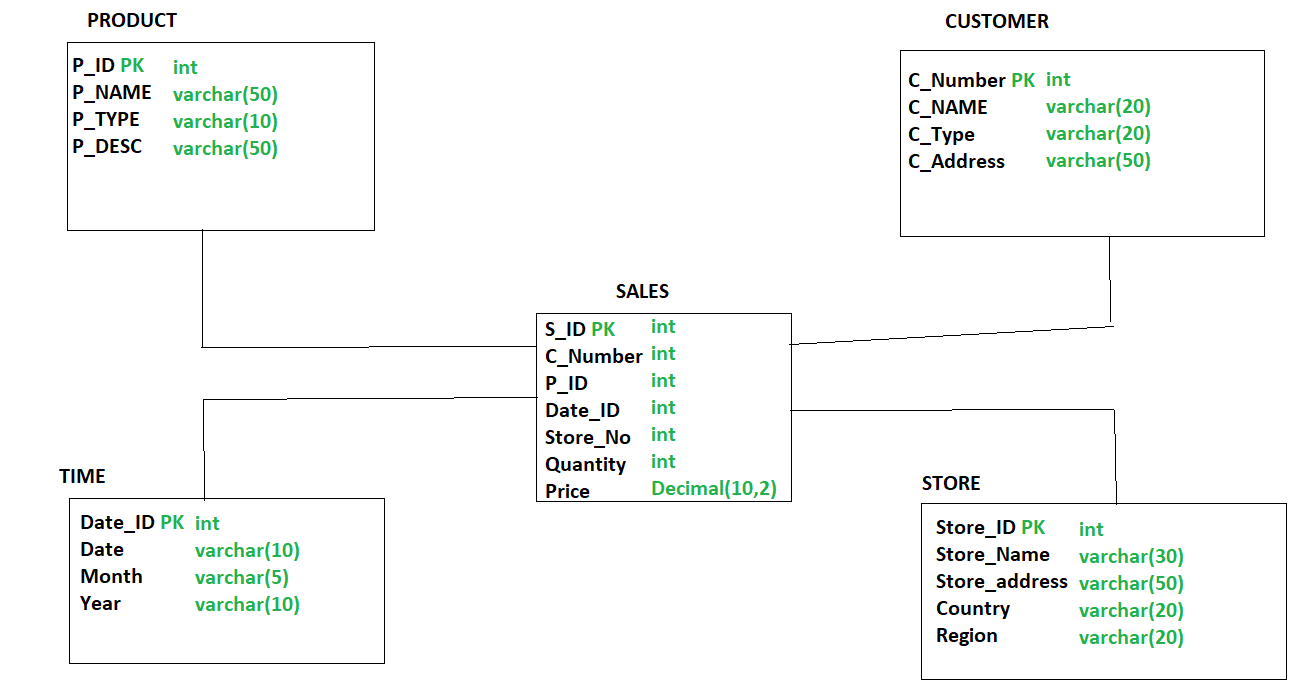
* Displays all the entities and attributes and relationship among them.
* Primary key for each entity is specified.
* Foreign keys for each entity if exists then specify.



**3.Physical Data Model**

Physical model defines how the model is physically exists in the system

* Displays all the table names and columns
* Displays foreign keys
* Displays relationship among all the tables and changed into FK.
* Entity becomes table.
* Attributes becomes column and its data types.



Dimensions and Facts

**Dimension:**

Dimension is nothing but descriptive characteristics ofattributes.

**Facts:**

Facts are measures or numbers.

**Types of dimensions**

1.SCD (Slowly Changing Dimension)

2.Confirmed dimension

3.Degenarated dimension

4.Junk Dimension

5.Static Dimension

6.Roll Playing dimension

7.Shrunken Dimension

**1.SCD (Slowly Changing Dimension)**

Dimension that changes slowly over a period of time rather than changing regularly is known as SCD.

**For ex: Location, City, Address, DOJ etc.**

**Types**

1.SCD Type1 (SCD1) – Current

2.SCD Type2 (SCD2) – Current as well as historical data.

3.SCD Type3 (SCD3) – Current as well as historical data but there is limited period of data we can store.

**Example: CUTOMER\_DETAILS**

**Consider a customer with name AMIT who is living in MUMBAI from 2018**

|  |  |  |  |
| --- | --- | --- | --- |
| CUST\_ID | CUST\_NAME | DOJ\_YEAR | LOCATION |
| 1 | AMIT | 2018 | MUMBAI |

**1.SCD Type1 (SCD1)**

Dimensions that replace old entry with new entry.

In SCD1 we can maintain only Current records.

In the above example on Location column if we apply SCD1 then if AMIT is changing his location to PUNE in year 2020, In SCD1 the above table changed as follows

Before Applying SCD1

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | SCD1 |
| CUST\_ID | CUST\_NAME | DOJ\_YEAR | LOCATION |
| 1 | AMIT | 2018 | MUMBAI |

`

After Applying SCD1

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | SCD1 |
| CUST\_ID | CUST\_NAME | DOJ\_YEAR | LOCATION |
| 1 | AMIT | 2020 | PUNE |

In this case, Previous entry which is treated as history, which will be lost in SCD1 and it maintains only current records.

**2.SCD Type 2 (SCD2)**

In SCD2 dimension we maintain current as well as historical records into the table.

To Identify current record and historical records from table we have two methods in SCD2

1.We must add columns like **START\_DATE** and **END\_DATE**.

2.We need to add column **ETL\_CurrentRecordFlag or ETL\_RecordFlag(True/False or Y/N or Current/Expired).**

* With the above **example** if AMIT is changing his location form MUMBAI to PUNE to INDORE in different years, then in **SCD2** we are maintaining all the history and current records and which will be identified clearly by adding **START\_DATE** and **END\_DATE** columns into table which shown below

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | SCD2 |  |  |
| CUST\_ID | CUST\_NAME | DOJ\_YEAR | LOCATION | START\_DATE | END\_DATE |
| 1 | AMIT | 2018 | MUMBAI | 1/1/2018 | 1/3/2020 |
| 1 | AMIT | 2020 | PUNE | 1/3/2020 | 31/05/2021 |
| 1 | AMIT | 2021 | INDORE | 31/05/2021 | NULL |

In the above table we can easily identify current records from table by writing simple SQL query

**SELECT \* FROM CUSTOMER\_DETAILS where END\_DATE is NULL**

* With the above **example** if AMIT is changing his location form MUMBAI to PUNE to INDORE in different years, then in **SCD2** we are maintaining all the history and current records and which will be identified clearly by adding **ETL\_CurrentRecordFlag** column into table which shown below

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | SCD2 |  |
| CUST\_ID | CUST\_NAME | DOJ\_YEAR | LOCATION | ETL\_CurrentRecordFlag |
| 1 | AMIT | 2018 | MUMBAI | Expired |
| 1 | AMIT | 2020 | PUNE | Expired |
| 1 | AMIT | 2021 | INDORE | Current |

In the above table we can easily identify current records from table by writing simple SQL query

**SELECT \* FROM CUSTOMER\_DETAILS where ETL\_CurrentRecordFlag = ‘current’**

**What is the need Primary Key in database table?**

* For good practice in database designing, we must maintain primary key in every table.
* Primary key is use to uniquely identifies each row/record in a table.

**Natural key**

* Primary key is made up of real data in table, that primary key is called as natural primary key.
* For example, In HR database, Employees table having **Eid** column which is unique and not null and we can call this real data because every employee should be identified by its **Eid**.
* Here we can make **Eid** as primary key in Employees table and this primary key is called as natural primary key.

|  |  |  |  |
| --- | --- | --- | --- |
| Eid | FirstName | Last Name | Dept |
| 1 | Kishor | Bhosale | Testing |
| 2 | Pavan | Patil | Dev |
| 3 | Amit | Chavan | Admin |

**Surrogate key**

* Some times in database table we cannot make primary key from real data.
* In this situation, we must add one artificial column in table which is having unique and not null values, and make this column as primary in table.
* This primary which is generated from artificial column is called as Surrogate key.

**Example:**

If we want to maintained history of employee table then **Eid** should be not the primary key column in table, because there are chances of duplicity in **Eid** column

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Eid | FirstName | Last Name | Dept | Date created |
| 1 | Kishor | Bhosale | Testing | 10/9/2021 |
| 2 | Pavan | Patil | Dev | 11/10/2021 |
| 3 | Amit | Chavan | Admin | 7/9/2021 |
| 1 | Kishor | Bhosale | Testing | 11/1/2022 |
| 2 | Pavan | Patil | Dev | 31/03/2022 |

* To resolve this problem of primary key in employee table, we must add column **ETL\_RECORD\_KEY** column which should be unique, not null values.
* And we can make **ETL\_RECORD\_KEY** as primary key in employee table, as this primary is also called as surrogate key.

**3.SCD Type3 (SCD3)**

In This Dimension New Field(S) / Columns are added to table to maintain the history but this change can accommodate only one change because of this reason it rarely used.

When the changes are maintained only once.

Example:

Before Applying SCD3

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | SCD3 |
| EMP\_ID | EMP\_NAME | EMP\_DOJ | EMP\_CITY |
| 1 | AMIT | 22/10/21 | PUNE |
| 2 | Rohit | 22/11/20 | MUMBAI |

After Applying SCD, after SCD3 applied it will add new column

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | SCD3 |  |
| EMP\_ID | EMP\_NAME | EMP\_DOJ | EMP\_CITY | OLD\_EMP\_CITY |
| 1 | AMIT | 22/10/21 | PUNE | NULL |
| 2 | Rohit | 22/11/20 | MUMBAI | NULL |

if Amit changes his location from PUNE to MUMBAI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | SCD3 |  |
| EMP\_ID | EMP\_NAME | EMP\_DOJ | EMP\_CITY | OLD\_EMP\_CITY |
| 1 | AMIT | 22/10/21 | MUMBAI | PUNE |
| 2 | Rohit | 22/11/20 | MUMBAI | NULL |

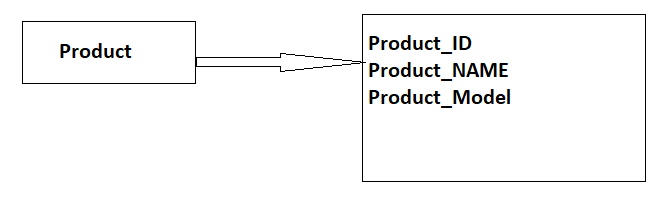
Again, if AMIT changes his location from MUMBAI to INDORE then

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | SCD3 |  |
| EMP\_ID | EMP\_NAME | EMP\_DOJ | EMP\_CITY | OLD\_EMP\_CITY |
| 1 | AMIT | 22/10/21 | INDORE | MUMBAI |
| 2 | Rohit | 22/11/20 | MUMBAI | NULL |

**2.Confirmed dimension**

A dimension table is used in multiple facts or a dimension which is used in multiple fact tables those dimensions considered as Confirmed dimension.

For Ex: If we have Product\_dimension Table which shown below



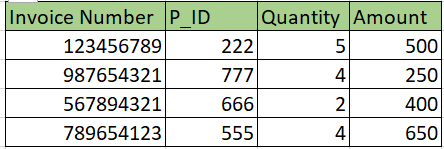
From the above dimension table, we are using **Product\_ID** dimension inside multiple fact tables i.e. Sales and Revenue but it meaning remains same so that we can called it is confirmed dimension.



**3.Dengenrated Dimension**

Degenerated dimension which is derived from fact table and does not have it’s dimension table.

**Example:**



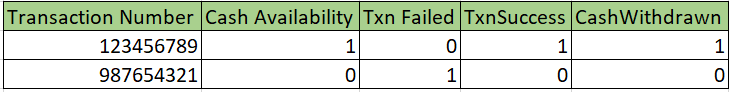
In the above example P\_ID comes from Product dimension table. Invoice number is a standalone attribute and has no other attributes/columns are associated with it.

**4. JUNK Dimension**

It is single table with combination of different and unrelated attributes/columns to avoid large number of FK.

Junk dimensions are used to reduce number of dimensions in the dimension model and reduce the number of columns in the fact table.

**Example:**



**5. Static Dimension**

Static dimension is nothing but constant. If this kind of dimensions are not extracted from Source but created in DWH context.

**Example:**

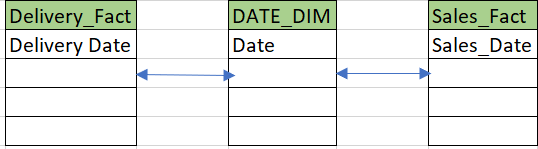
Gender is constant i.e., MALE and FEMALE.

**6.Roll Playing Dimension**

Dimension which is often used for multiple purpose within the same database are called Roll Playing dimension.

**Example:**

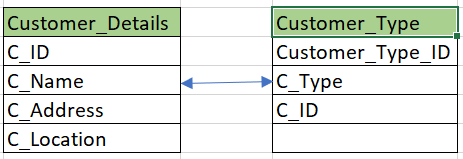
A fact table may include FK for both sales and delivery table.



**7.Shrunken Dimension**

A shrunken dimension is subset of another aspect.

**Example:** In the below example **customer Type** is derived from **Customer\_Details i.e customer\_Type**



**Facts:**

Facts are nothing but measures or we can say numeric values.

Fact is numeric value which have impact on business process.

**Ex: sales, quantity, profit etc.**

**BOA (Bank Of America)**

**DIM\_BOA\_CUSTOMER\_DETAILS**

**FACT\_BOA\_REVENUE**

**Types of facts**

1.Additive Facts

2.Semi-Additive facts

3.Non-Additive Facts

**1.Additive Facts**

Additive Facts are those facts which can be summed up through all the Dimension keys from fact table.

The fact which is added across all the dimensions is known as additive fact.

**Example:**

FACT\_SALES

P\_ID, C\_ID, S\_ID, D\_ID, Quantity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P\_ID (DMK) -FK | C\_ID(DMK) -FK | S\_ID(DMK) -FK | D\_ID(DMK) -FK | Quantity (Fact) |
| 111 | 5678 | 6543 | 20220921 | 30 |
| 222 | 7865 | 7766 | 20220920 | 50 |
| 111 | 5678 | 6543 | 20220919 | 40 |
| 222 | 7865 | 7766 | 20220921 | 30 |

Q. What is the total quantity sold of product 222?

Q. What is total quantity sold for customer 5678?

Q. What is total quantity sold on 21st sept 2022?

Q. What is total quantity sold for store 7766?

**2.Semi-Additive facts**

Semi-Additive Facts are those facts which can be summed of for some of the dimension keys from fact table.

Facts which can be added across some of the dimension is known as semi-additive facts.

**Example: Fact\_Current\_Balance**

|  |  |  |
| --- | --- | --- |
| **Date** | **Account** | **Current\_Balance** |
| **20210913** | **1111288785** | **1000** |
| **20210913** | **1111288786** | **500** |
| **20210913** | **1111288787** | **700** |
| **20210914** | **1111288785** | **900** |
| **20210914** | **1111288786** | **1000** |
| **20210914** | **1111288787** | **800** |

**Current Balance** is semi additive fact, It makes sense to add them for particular date entries so that we can get total current balance for the day and which will be useful along with date but when we add w.r.t accounts and total current balance for accounts is some number then it wont make any sense adding current balance w.r.to accounts.

**For Example:**

If you have number of items in warehouse for each day, You can sum up the items for each day(total items in the warehouse for each day) but it wont make sense to sum up in the year.

**3.Non-Additive Facts**

Non-Additive fact are those facts which cannot be summed up for any of the dimension keys from the fact table.

The fact which cannot be added across any of the dimension keys.

Example: FACT\_CURRENT\_BALANCE

SAF NAF

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Account** | **Current Balance** | **Profit (%)** |
| **20210913** | **1111288785** | **1000** | **70** |
| **20210913** | **1111288786** | **500** | **40** |
| **20210913** | **1111288787** | **700** | **60** |
| **20210914** | **1111288785** | **900** | **80** |
| **20210914** | **1111288786** | **1000** | **60** |
| **20210914** | **1111288787** | **800** | **50** |

If we will add profit for account 1111288785 for 13th and 14th sept i.e., 70% + 80% and saying profit is around 150%.

It does not make sense to add profit along with dates or account level, so we can say profit is non-additive fact.

**Fact Tables**

1.Transaction Fact Table

2.Snapshot Fact Table

3.Accumalated Fact Table

4.Fact Less Fact Table.

**1.Transaction Fact Table**

Transaction Fact Table is a basic approach to operate business.

These fact tables represent an event that occurs at any point of time.

Always in Transaction fact Table we are mostly observed Additive Facts.

**Example:**

|  |  |  |  |
| --- | --- | --- | --- |
| Account | TransactionID | AmountPaid | DateID |
| 111288785 | 123456789 | 10 | 20210913 |
| 111288785 | 123456790 | 20 | 20210913 |
| 111288785 | 123456791 | 10 | 20210913 |
| 111288785 | 123456789 | 10 | 20210913 |

**2.Snapshot Fact Table**

It will store current state(information) at regular interval of time (Hours, days, week, months etc.)

Its best use to analyses business performance at fixed interval.

**Example:** Fact\_Hospital\_Beds\_Availability

|  |  |  |  |
| --- | --- | --- | --- |
| **H\_DEPT\_ID** | **Total\_Beds** | **Hourwise\_occupied\_beds** | **Hourwise\_Free\_beds\_update** |
| **111** | **150** | **75** | **75** |
| **222** | **200** | **80** | **120** |
| **333** | **150** | **70** | **80** |

In Hospital we are maintaining the beds availability for patients on hour basis and that kind information if we are updating in particular table then that table contains every hour information about the beds i.e., it takes snap of hour basis and maintain in table that table is nothing but Snapshot fact table.

**3.Accumalated Fact Table**

It will help you to find, what happened over a period of time.

This table will maintain information about a complete life cycle of a product or application

**Example:**

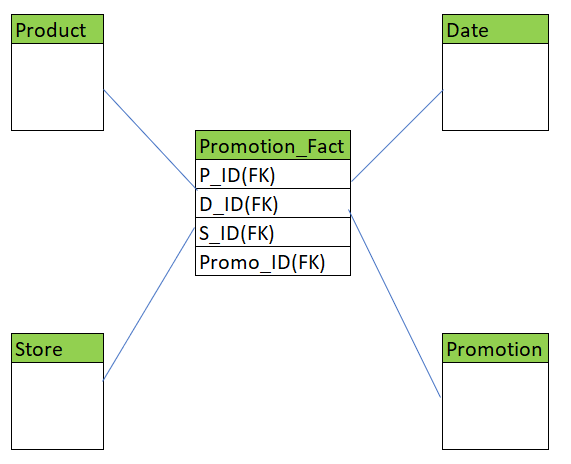
Flipkart product from order to till its delivery and return, is one complete cycle and that information is stored in a table and that table is nothing but accumulated fact table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Order\_Date | Bill\_Date | Shipping\_Date | Arrival\_Date | Delivery\_Date | Return\_Date |
| 22/09/22 | 22/09/22 | 23/09/22 | 25/09/22 | 26/09/22 | 03/10/22 |
| 22/09/21 | 22/09/21 | 23/09/221 | 25/09/21 | 26/09/21 | 03/10/21 |

**4.Fact Less Fact Table**

A fact less fact table is fact table that does not have any measures or original facts instead it has all the dimension keys in that table.

**Example:**

In this example we are promoting a product and for that we are maintaining a fact table called promotion fact table which contains only dimension keys in it

**Data Schemas**

**(In Real time based data schemas we are able to get document called DSD (Design Specification Document) or HLD (High Level Design Document) or HLSD (High Level Specification Design Document)) and It is prepared by Principle Architect/ Data Base Designer.**

A database schema defines its entities and relationships among them.

Database Schema is a descriptive detail of the database and which can be implemented by means of Schema diagram.

1.Star Schema

2.Snowflake Schema

3.Fact Constellation Schema.

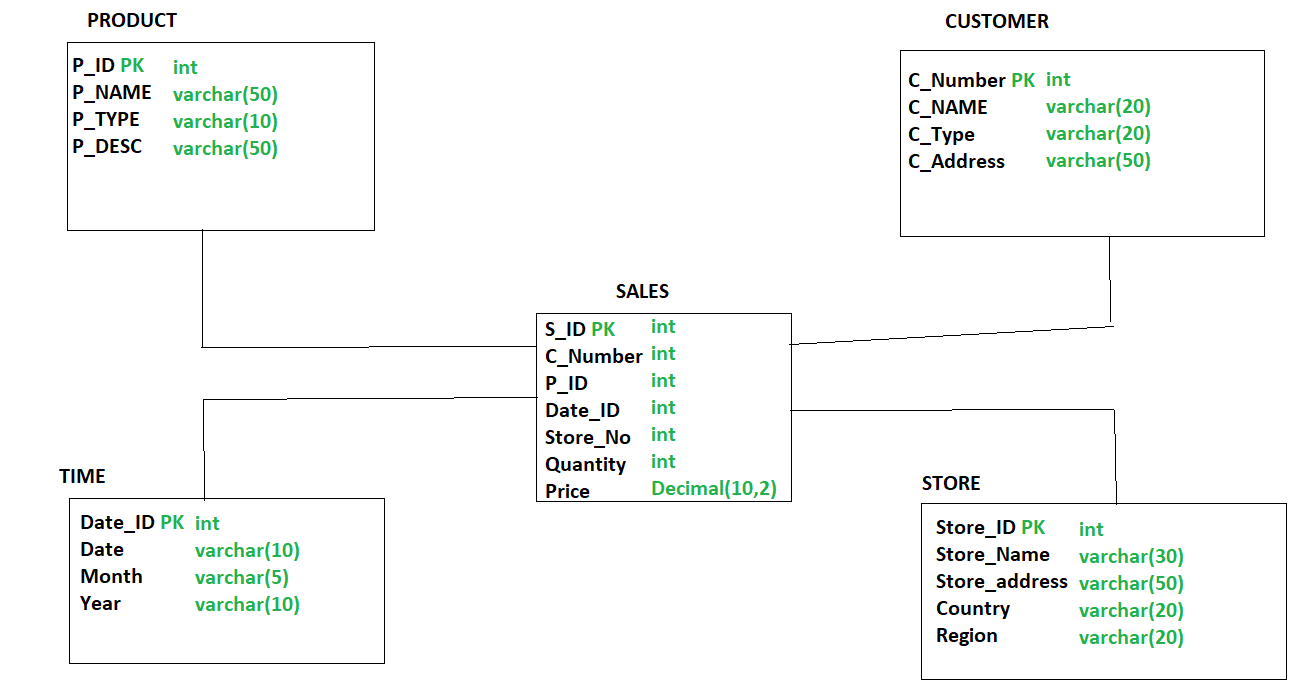
**1.Star Schema**

Star Schema consists of fact table at the center and all the dimensions around it.

Fact table is usually a sum of all dimensions and dimension is single entity.

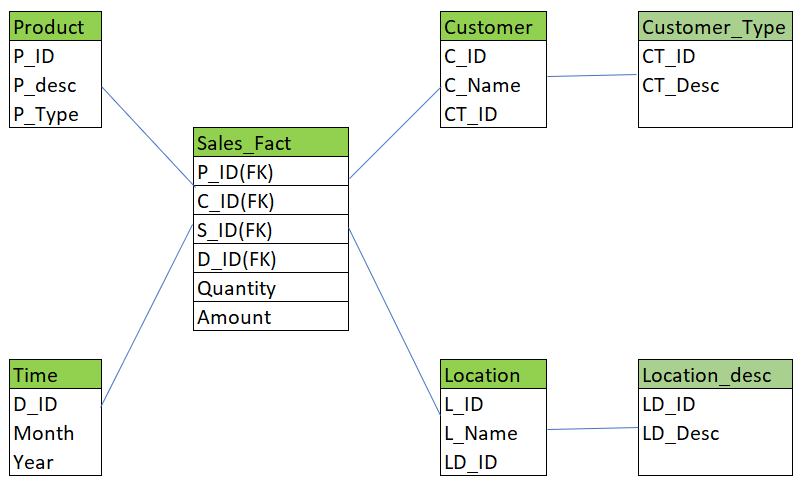
A primary key from dimension table is represented as foreign key in fact table.

**Example:**



**2.Snowflake Schema**

This is extension of star schema, where each point of start or dimension is divided into more granular level.



**3.Fact Constellation (Galaxy) Schema**

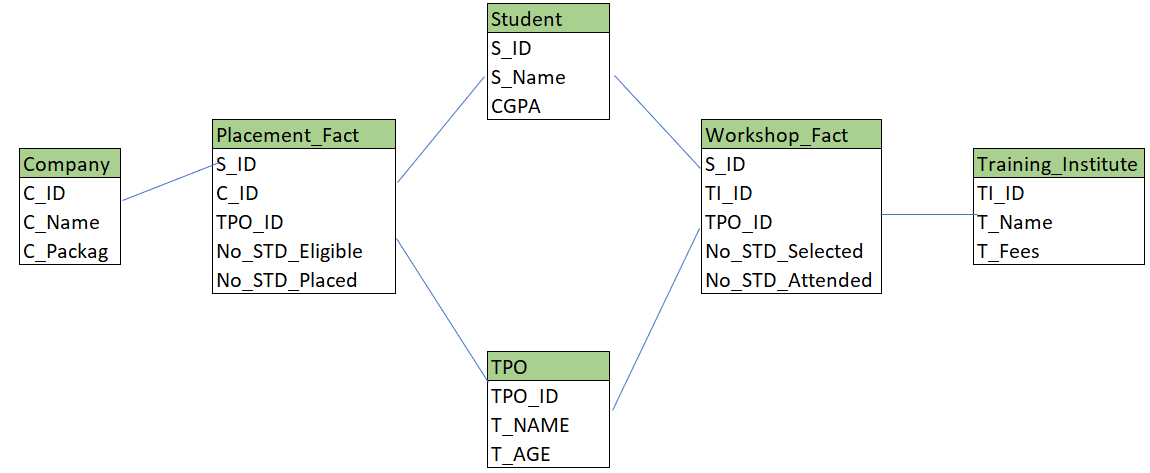
Fact Constellation is Schema for representing multidimensional model.

It is collection of multiple fact tables having common dimension tables.

It is collection of several star schemas and hence, it is called as Galaxy schema.

It is one of the widely used schemas for Data warehouse designing and it is more complex than star and snowflake schema.

**Example:**



**SME: Subject Matters Expert – This person will become Principle Architect – He is aware about the complete domain knowledge, like Banking , Telecom, insurance etc.**

**Normalization and Denormalization**

Normalization is a process of eliminating or removing the redundant (Repetitive) data and storing the related information in a table.

The key points of Normalization are as below:

* Eliminating Redundant Data
* Faster Update
* Improve Performance
* Performance in Indexes

Below are the different Normal Forms

**First Normal Form:**

If the table is said to be in the 1st Normal Form it should follow the below rules

* Each cell must have one value. Eliminating Duplicate Columns
* Create separate table for the group of the related data and each row should be identified by Primary Key.

Let us take an example here:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Department** | **Phone Number** | **Salary** | **TAX** |
| Name 1 | Computers | 12354588, 6887890, 123456 | 4000 | 40 |
| Name 2 | Electronics | 12345678 2345666,7896543333 | 5000 | 50 |
| Name 3 | Civil | 4567890 | 3000 | 30 |

In the above table we see that there are different phone numbers for the single Name and we have to remove these duplicates by uniquely identifying each of them and giving a unique identification by giving them the Primary Key.

Now the below table is redesigned such that it is in the First Normal Form.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID(PK)** | **Name** | **Department** | **Phone Number** | **Salary** | **TAX** |
| 1 | Name 1 | Computers | 12354588 | 4000 | 40 |
| 2 | Name 1 | Computers | 6887890 | 4000 | 40 |
| 3 | Name 1 | Computers | 123456 | 4000 | 40 |
| 4 | Name 2 | Electronics | 12345678 | 4000 | 40 |
| 5 | Name 2 | Electronics | 2345666 | 4000 | 40 |
| 6 | Name 2 | Electronics | 789654333 | 4000 | 40 |
| 7 | Name 3 | Civil | 4567890 | 3000 | 30 |

**Second Normal Form:**

If the table is said to be in the 2nd Normal Form it should follow the below rules

* It should satisfy the 1st Normal Form.
* Separate the Columns, values are duplicated in each row should be placed in the separate Table.
* Create the relationship between the tables.

Here in the above table, we see that the Name and the department columns are duplicated and in order to handle this we need to maintain the duplicates in the different table as below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMPID(PK)** | **Name** | **Department** | **Salary** | **TAX** |
| 1 | Name 1 | Computers | 4000 | 40 |
| 2 | Name 2 | Electronics | 4000 | 40 |
| 3 | Name 3 | Civil | 3000 | 30 |

|  |  |  |
| --- | --- | --- |
| **ID(PK)** | **EMPID(FK)** | **Phone Number** |
| 1 | 1 | 12354588 |
| 2 | 1 | 6887890 |
| 3 | 1 | 123456 |
| 4 | 2 | 12345678 |
| 5 | 2 | 2345666 |
| 6 | 2 | 789654333 |
| 7 | 3 | 4567890 |

Here in these tables above EMPID is treated as the primary Key for the First Table and the Foreign Key for the Second Table.

**Third Normal Form:**

If the table is said to be in the 3rd Normal Form it should follow the below rules

* It should satisfy the 2nd Normal Form.
* Separate the Columns that are not dependent on the primary key of the table.

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPID(PK)** | **Name** | **Department** | **Salary** |
| 1 | Name 1 | Computers | 4000 |
| 2 | Name 2 | Electronics | 4000 |
| 3 | Name 3 | Civil | 3000 |

|  |  |  |
| --- | --- | --- |
| **ID(PK)** | **EMPID(FK)** | **Phone Number** |
| 1 | 1 | 12354588 |
| 2 | 1 | 6887890 |
| 3 | 1 | 123456 |
| 4 | 2 | 12345678 |
| 5 | 2 | 2345666 |
| 6 | 2 | 789654333 |
| 7 | 3 | 4567890 |

|  |  |
| --- | --- |
| **Salary** | **TAX** |
| 4000 | 40 |
| 3000 | 30 |

**De-Normalization:**

De-normalization is a database optimization technique in which we add redundant (repetitive) data to one or more tables. This can help us avoid costly joins in a relational database.

Note that de-normalization does not mean not doing normalization. It is an optimization technique that is applied after doing normalization.

**Example:**

We have two table **students** and **branch** after performing normalization.

The student table has the attributes Roll\_no, stud-name, age, and Branch\_id.

|  |  |  |  |
| --- | --- | --- | --- |
| Roll\_No | Student\_Name | Age | Branch\_ID |
| 11 | Stevan | 23 | 1 |
| 12 | Mark | 22 | 2 |
| 13 | Nishem | 25 | 1 |
| 14 | Taylor | 21 | 3 |
| 15 | Kane | 26 | 4 |

Additionally,

the branch table is related to the student table with branch\_id as the student table's foreign key.

|  |  |  |
| --- | --- | --- |
| Branch\_ID | Branch\_Name | Total\_students |
| 1 | EC | 150 |
| 2 | CSE | 100 |
| 3 | Civil | 60 |
| 4 | Mech | 80 |

A JOIN operation between these two tables is needed when we need to retrieve all student names as well as the branch name.

Suppose we want to change the student’s name only, then it is great if the table is small. The issue here is that if the tables are big, joins on tables can take an excessively long time.

In this case, we'll update the database with de-normalization, redundancy, and extra effort to maximize the efficiency benefits of fewer joins. Therefore, we can add the branch name's data from the Branch table to the student table and optimizing the database.

**ETL**

**ETL – Extract Transform and Load**

It is process, which defines how data will load from different sources to target or Data ware house by applying multiple transformations.

Data is extracted from various OLTP database and transformed to match data warehouse schema and loaded into data warehouse database.

**In General**

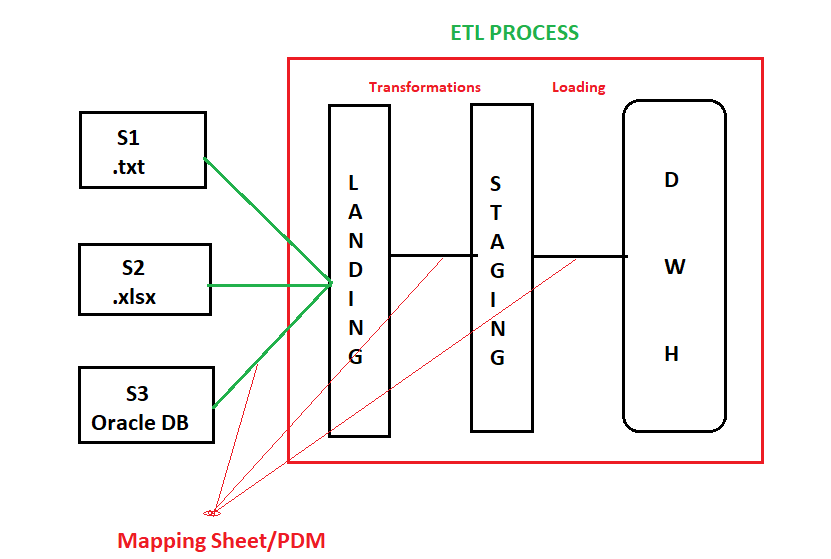
**Source Data 🡪 Staging Area 🡪 DWH 🡪 DM (Data Mart)**

**In real time operation layer may vary from project to project as per requirement**

**SOURCE DATA 🡪 LANDING AREA 🡪 STAGING AREA 🡪 DWH**

**ETL Architecture**

In each ETL tool refers the same architecture and ETL terminology will not change for any ETL tool, instead it will be same for each ETL tool available in market.



**Mapping sheet /PDM (Progressive Data Mapping):**

It is document prepared by BA (Business Analyst), It contains all the information about, how the data is transferred from Source to target.

It includes all the table structure and column to column mapping information such as Table Name, data types, size and if any transformation is applied on any column.

If in Target, new column is created by joining two tables then that JOIN information is also maintained in mapping sheet.

We can map data in two ways i.e.,

* 1:1 mapping sheet

Ex: In source Column name is FIRST NAME and In Target it is FIRSTNAME along with data type.

* Many: 1 Mapping sheet

Ex: in Source if we have FirstName, Middle Name, Last Name and In Target It is derived as FULLNAME

**1st Phase**

**Source to Landing**

Extracting data from Source to landing and stored in structure format.

This structure format information will be maintained in mapping sheet and based upon that only data will be extracted from source to Landing.

**Landing:**

This Layer is just a replica from source data but it should convert all the source data into structure format as per mapping sheet.

Landing is the first stage of ETL process where we found data is in structure format.

**2nd Phase**

**Landing to Staging**

We are extracting data from Landing Layer to staging layer.

While extracting data from Landing layer to Staging layer again they will follow mapping sheet which is prepared for Landing to Staging.

**Staging Layer:**

This layer is used to hold data and perform cleansing and merging before loading data into DWH.

Transforming the data involve multiple steps

* Cleaning: For example, in Source column Gender in that we have data **Female** and **Male** but we need same data in target as **‘F’** and **‘M’**, if this conversation is performed and same has been mentioned inside mapping sheet i.e., Cleaning.
* Filtering: selecting only certain column in target table

For example: suppose if we have FirstName, Middle Name and Last Name column in Source but in target selecting only First Name and Last Name.

Suppose in source we have column called **Region** and in that we have data like **North, East, South and North** but in target we need a data only from **south and north** then by filtering we loads that data into target.

* Splitting: A column split into multiple columns.

For Example: If we have **Full Name** column in source and in target, we need **FirstName and Last Name** Column then we use split and convert.

* Data Merging: It is process of integrating data from multiple sources of similar structure or dissimilar structure into a single output.

**3rd Phase**

**Staging to DWH**

We are extracting data from staging layer and load it into DWH

**DWH**

**Loading**

It is process of writing data into Data Warehouse or Target Database.

**Documents Required for ETL Testing**

**1.SRS** (System Requirement Specification)**/BRS** (Business Requirement Specification)**/FSD** (Functional specification Document).

Format: Word Document (.docs)

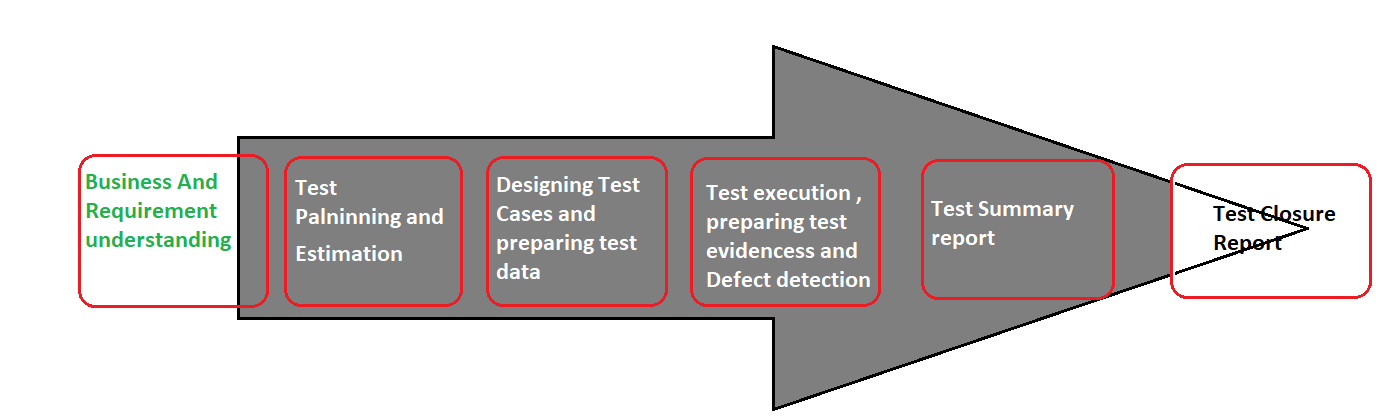
**2.Mapping Sheet**

Format: Excel document (.xlsx)

**3.HLSD** (High Level Specification Design)**/HLD** (High Level Design)**/DSD** (Design Specification Document).

Format: Power point Presentation (.ppt/.pptx)

**ETL Life Cycle:**



**1.Business and requirement understanding/ analysis:**

Business requirement document and data mapping document is prepared and reviewed by Business analyst and understand the requirement and provide a walkthrough session to both developer and tester.

These documents help us to understand

* Structure and usage of a data
* Clarifies data relate issues
* Analyse mappings and data transformation logics

After documentation part, validate requirement with business analyst in order to make sure, their should not be any gap between requirement and understanding.

**2. Test Planning and Estimation:**

In this phase estimation involves time and effort required for testing process to complete, based on estimation test planning will be done.

**3. Designing Test cases and preparing test data**

In this phase ETL test engineer will identify the scenarios and design the test cases with all possible test scenarios considering all the positive and negative inputs.

During this phase if source data is not available to test then, sometimes ETL tester will prepare dummy data for testing but in organization we have team called TDM (Test Data Management) Team will be responsible to prepare data.

**4. Test Execution, Preparing test evidences and Defect Detection.**

Once test case design is completed and revived by BA and providing a sign-off then, ETL tester will start test execution and if test cases are executed as per requirement, then will pass the test cases, and will prepare testing evidences for the same for justification because as ETL tester we need to provide proof of testing so that testing evidences will be required.

During this phase if we found an any issue in any of the test cases then we mark them as fail and raise defect and assigned back to respective developer, who has developed that functionality.

**5. Test summary report (TSR)**

In this phase, after completion of testing we need to prepare Test summary report, Any one of the team member is going to prepare TSR, which Includes complete summary about testing i.e., How many test cases we have prepared, how many of them passed and how many them failed and how many defects have been raised and how much time it has been taken to complete the testing.

To prepare TSR, a standard template will be provided with inclusion of all the things mentioned above.

**6. Test closer report.**

In this phase we will take complete sign-off for the release from client or stake holder.

During this phase they will ask you sometime to provide complete walkthrough about your testing for a particular release and senior team member will provide walkthrough to client and explain how we have performed testing and what are the different scenarios we have covered.

**ETL Packages and transformations:**

**Requirement:** SSIS Tool

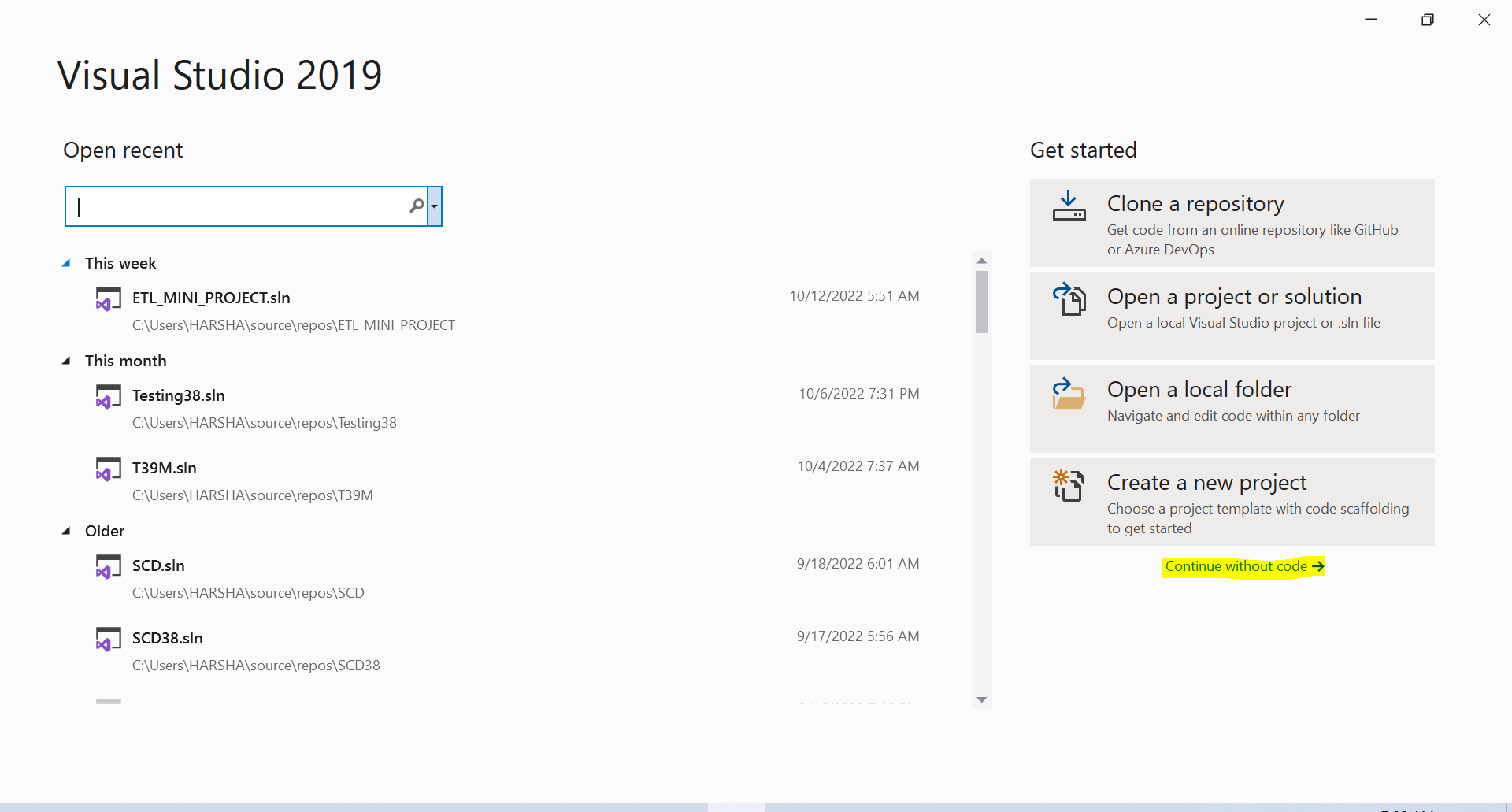
**Responsible Person of Package Creation:** ETL Developer

**SSIS Tool:**

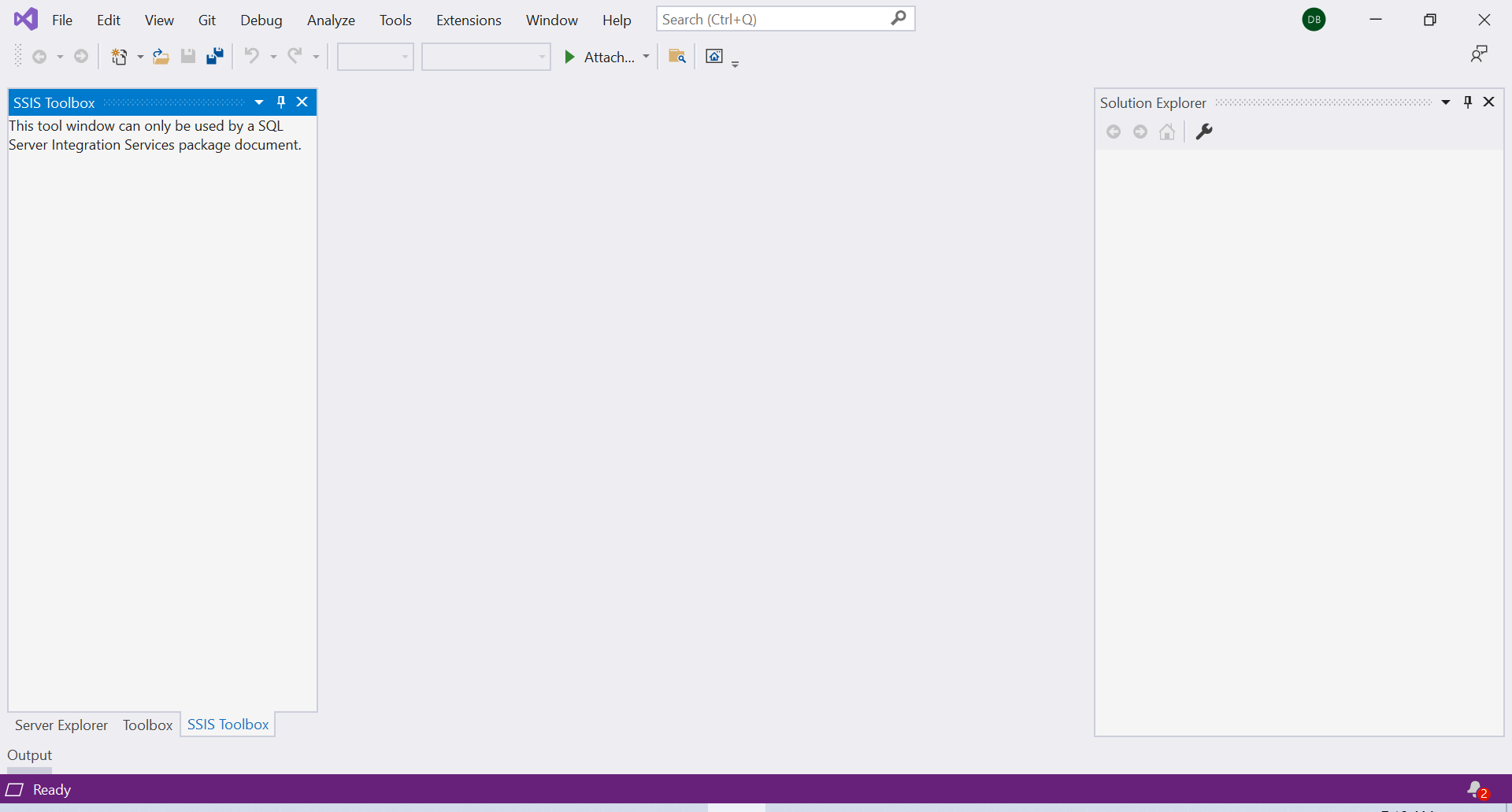
It is a data warehousing tool used for data extraction, loading the data into another database, transformations such as cleaning, aggregating, merging data, etc.

**How to open SSIS ETL Tool**

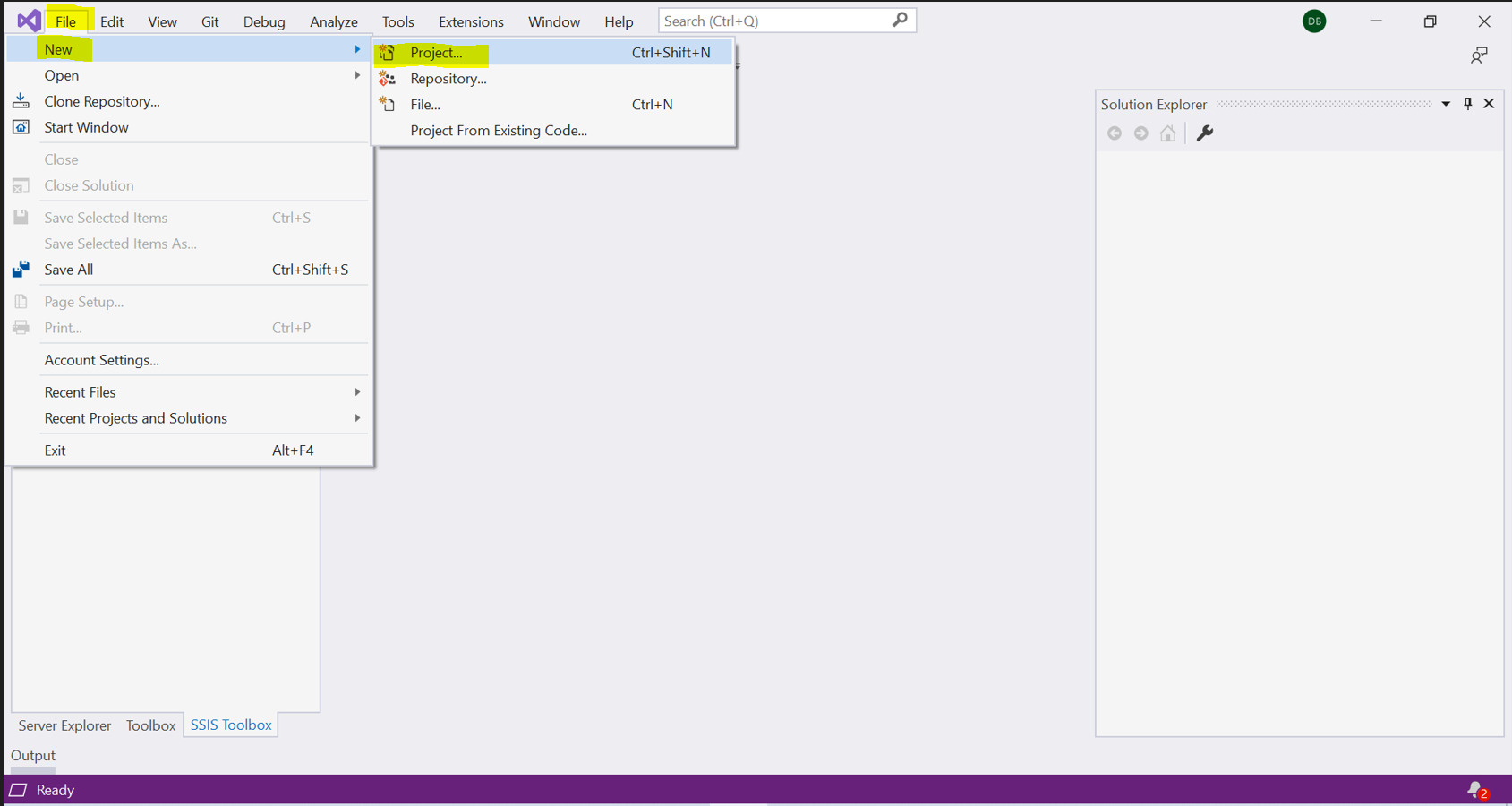
1.Go to start menu type Visual studio and click



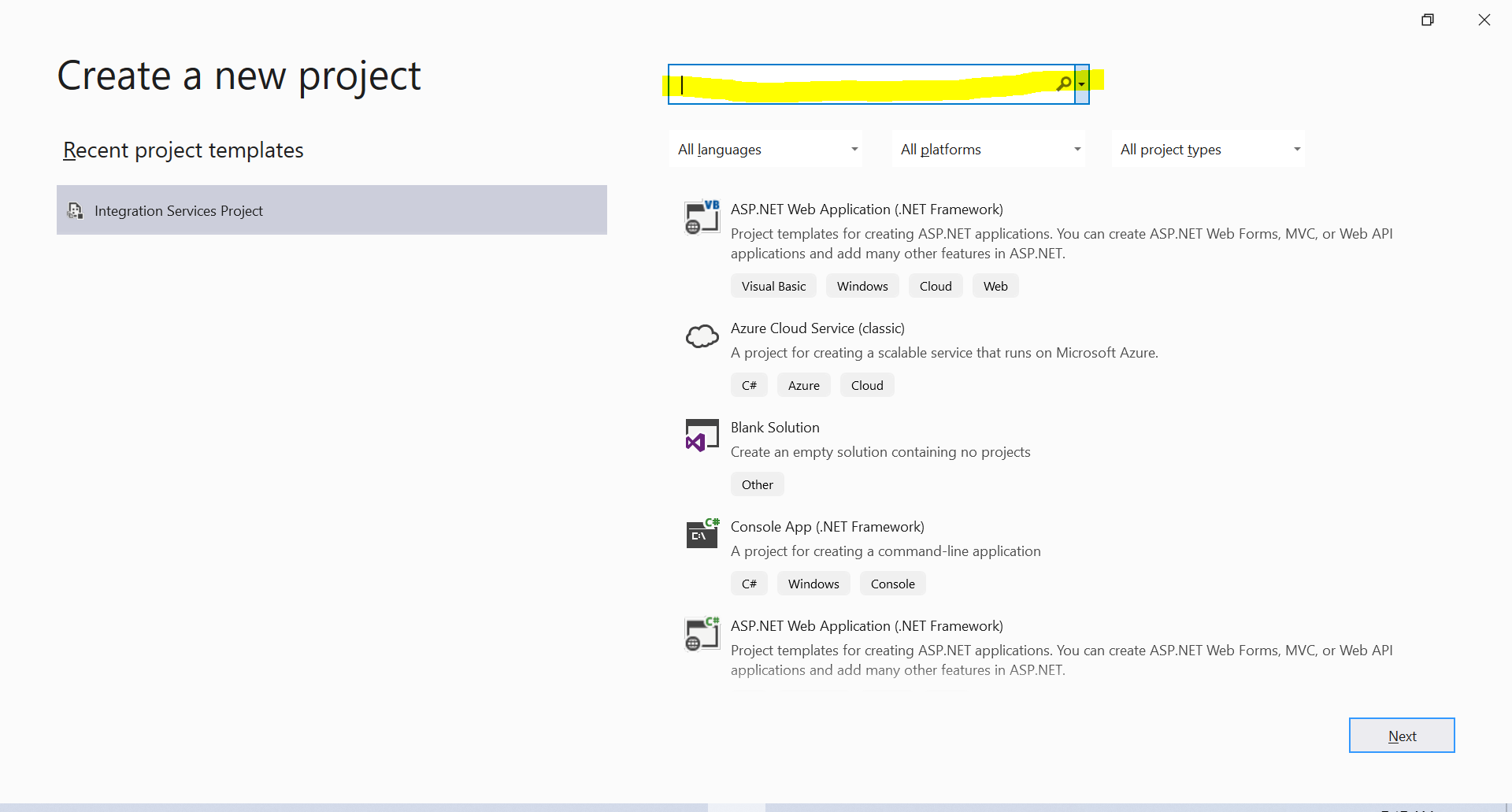
2. Click on Continue without code which is highlighted in yellow colour in above screenshot then below window will appear



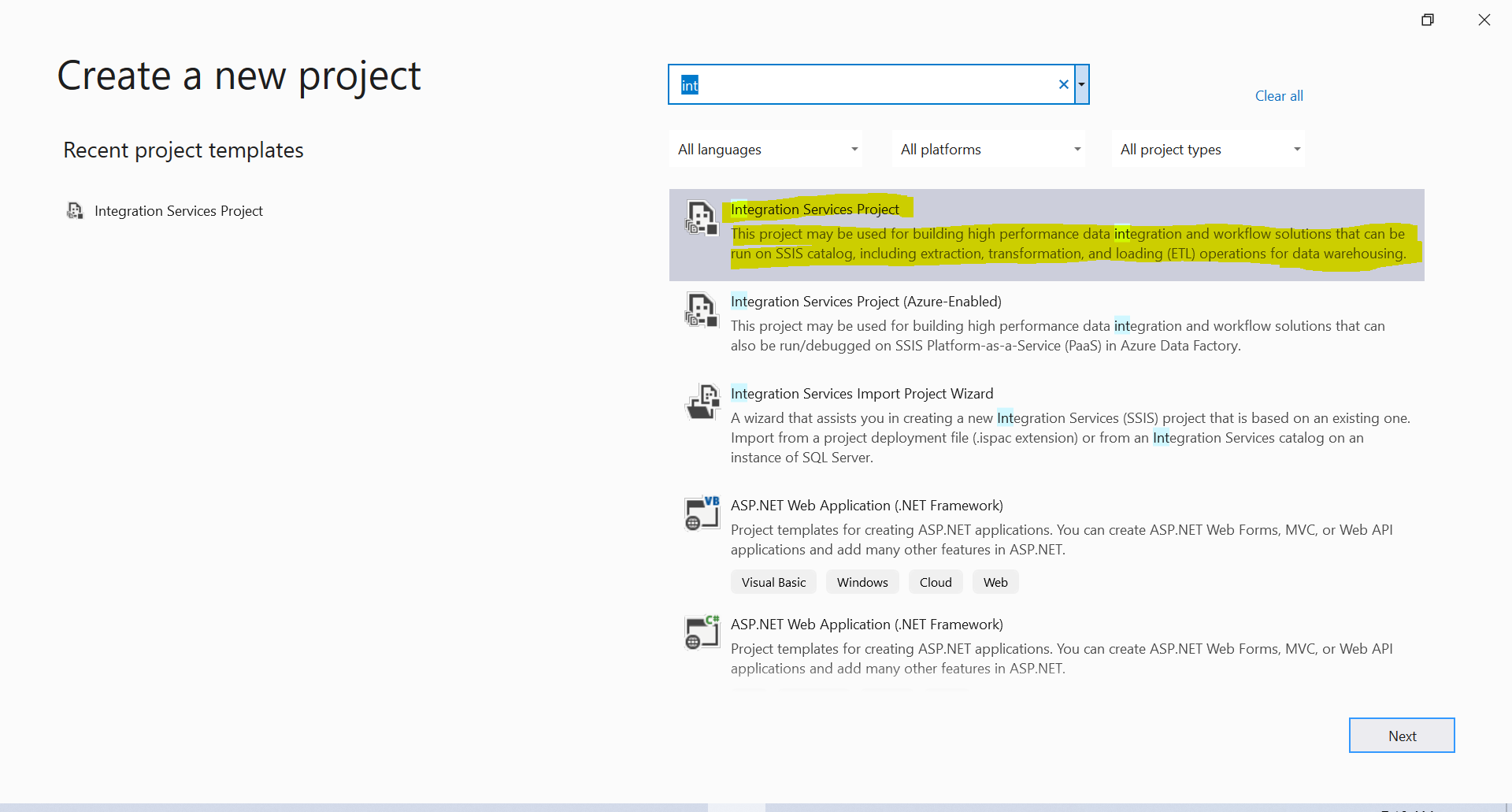
3.Go to file and create click on new and create project



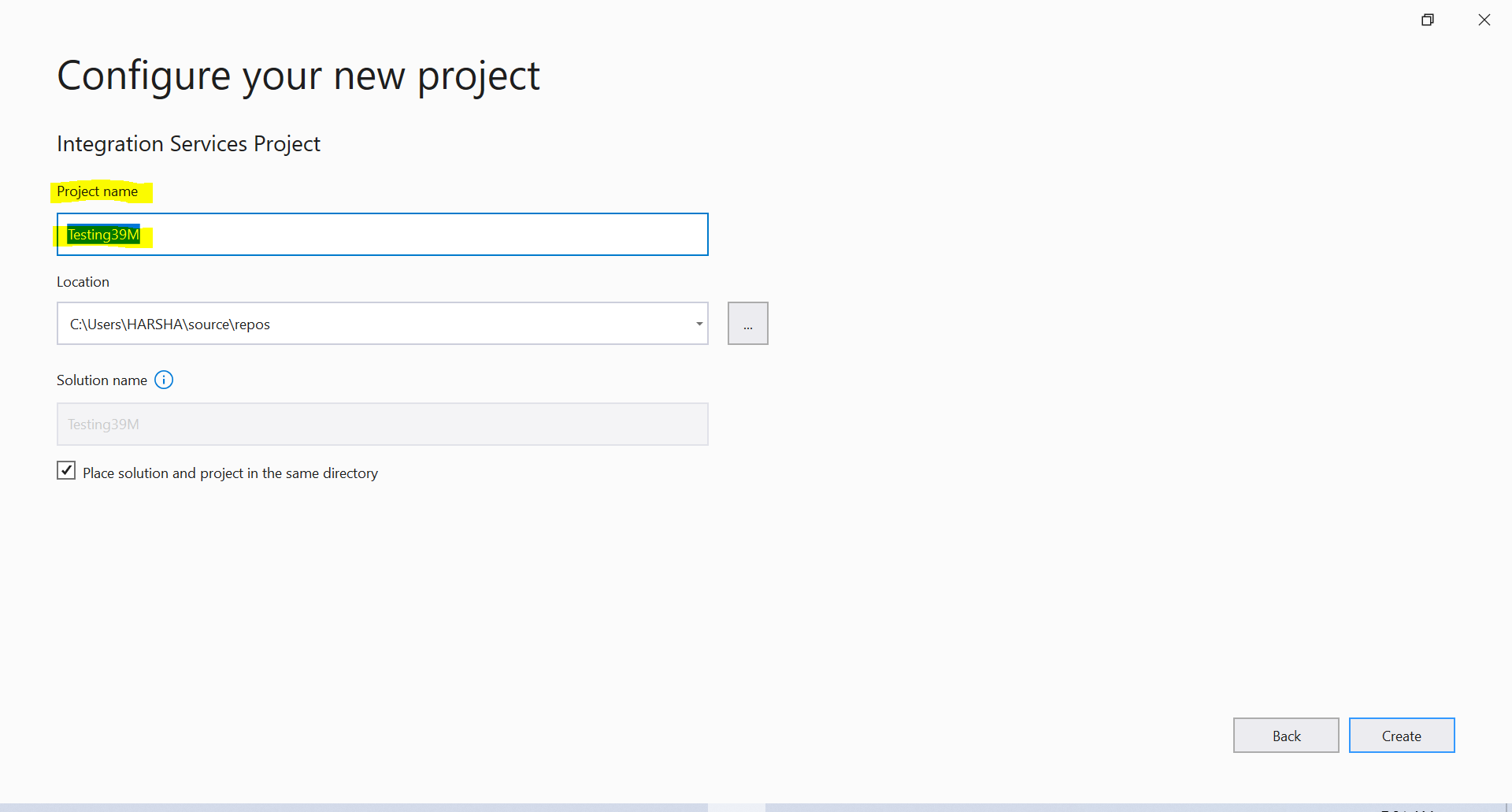
4. Type Integration services in yellow highlighted search window



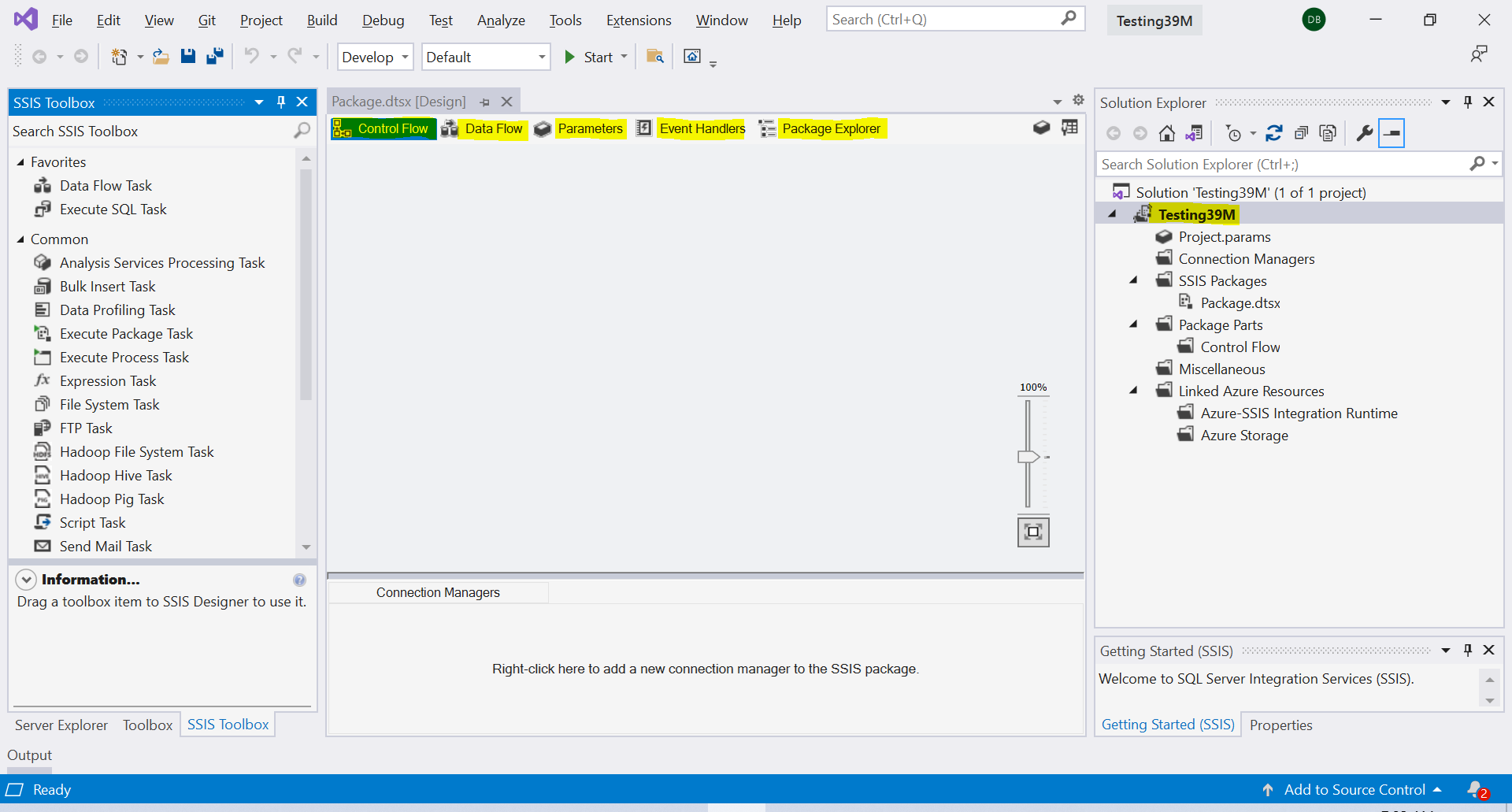
then integration services project will appear and click on that then it will allow us to create ETL project and click Next at the bottom



5.Write your project name in project name window which is highlighted in Yellow and click on create button



6. Project is created and all the SSIS ETL tool tabs are available and which is highlighted in Yellow



In the above screen SSIS tabs are

* Control Flow
* Data Flow
* Parameters
* Event Handlers
* Project Explorer

**SSIS Date/Time Data Types:**

* **DT\_BOOL:** It is a 1-bit Boolean Value
* **DT\_BYTES:** It is a binary data value and the length is variable, the maximum length is up-to 8000 bytes.
* **DT\_CY:** It is a currency value and this data type is an eight-byte signed integer with a scale of 4 and a maximum precision of 19 digits
* **DT\_DATE (Format: yyyy-mm-dd):** It is a data structure that consists of year, month, day, hour, minute, seconds, and fractional seconds. The maximum scale of fractional seconds is 7 digits
* **DT\_DBDATE:** A date structure that consists of year, month, and day.
* **DT\_DBTIME (Format: hh:mm: ss):** A time structure that consists of hour, minute, and second.
* **DT\_DBTIME2 (Format: hh:mm: ss[.fffffff]):** A time structure that consists of hour, minute, second, and fractional seconds. The maximum scale of fractional seconds is 7 digits
* **DT\_DBTIMESTAMP (Format: yyyy-mm-dd hh:mm: ss[. fff]):** A timestamp structure that consists of year, month, day, hour, minute, second, and fractional seconds. The maximum scale of fractional seconds is 3 digits
* **DT\_DBTIMESTAMP2 (Format: yyyy-mm-dd hh:mm: ss[.fffffff]):** A timestamp structure that consists of year, month, day, hour, minute, second, and fractional seconds. The maximum scale of fractional seconds is 7 digits
* **DT\_DBTIMESTAMPOFFSET (Format: yyyy-mm-dd hh:mm: ss[.fffffff] [{+|-} hh:mm]):**A timestamp structure that consists of year, month, day, hour, minute, second, and fractional seconds. The maximum scale of fractional seconds is 7 digits
* **DT\_DECIMAL:** An exact numeric value with a fixed precision and a fixed scale. This data type is a12-byte unsigned integer with a separate sign, a scale of 0 to 28, and a maximum precision of 29
* **DT\_FILETIME (Format: yyyy-mm-dd hh:mm:ss:fff):** A 64-bit value that represents the number of 100-nanosecond intervals maximum scale of fractional seconds is 3 digits
* **DT\_GUID:** It is a globally unique identifier (GUID).
* **DT\_I1:** It is a one-byte, signed integer.
* **DT\_I2:** It is a two-byte, signed integer
* **DT\_I4:** It is a four-byte, signed integer.
* **DT\_I8:** It is an eight-byte, signed integer
* **DT\_NUMERIC:** An exact numeric value with a fixed precision and scale. This data type is a 16-byte unsigned integer with a separate sign.
* DT\_R4: It is a single-precision floating-point value. --real
* DT\_R8: It is a double-precision floating-point value --float
* **DT\_STR:** It is a null-terminated ANSI/MBCS character string with a maximum length of 8000 characters --Varchar
* **DT\_UI1:** It is a one-byte, unsigned integer
* **DT\_UI2:** It is a two-byte, unsigned integer
* **DT\_UI4:** It is a four-byte, unsigned integer.
* **DT\_UI8:** It is an eight-byte, unsigned integer.
* **DT\_WSTR:** It is a null-terminated Unicode character string with a maximum length of 4000 characters –Nvarchar – one character as 2bytes
* **DT\_IMAGE:** It is a binary value with a maximum size of 2^31 -1 byte
* **DT\_NTEXT:** It is a Unicode character string with a maximum length of 2^30-1 characters
* DT\_TEXT: An ANSI character string with a maximum length of 2^31-1 characters

**Transformation:**

**1.Data Conversion:**

This type of transformation is used to convert source data into user defined data types or required target data types which is always defined in ETL mapping sheet.

As per mapping sheet we can change source column data type into target data type and we can load.

**2.Conditional Split**

This Transformation is used to Split the source data based upon condition and load it into different targets.

For example:

In Source data we have Segment Column and in that we have value Consumer, Corporate and Home Office based on three values we can create three target tables and load that data segment wise.

**3.Sort transformation**

Sort Transformation is an SSIS component, which is used to sort data based on columns either in ascending or descending order.

**4.Merge Transformation**

Merge Transformation is an SSIS component that can combine two sorted inputs into a single output.

For Example, if we have source in Excel and DB then we can merge and both data into single target.

**5.Merge Join transformation**

Merge Join Transformation is an SSIS component that can join two sorted inputs based on inner join, left outer join and Full outer Join, and produce a single output.

Q. Take same data in two different formats i.e., .xlsx (Excel sheet) and .txt (Flat File) and apply merge join transformation and load data into a single Target in Data Base.

**6.Aggregate Transformation**

Aggregate Transformation is a SSIS component, which is used to aggregate source data by using aggregate function like Count, Sum, Min, Max, Avg and we can group by using a particular groups and load data into target.

**7. Multicast Transformation**

Multicast transformation is a SSIS component which is used to load source data into multiple targets.

**For Example:** If source is excel file and it is newly added source in our application then we need to add the same source in structure format and want place in all the layers of ETL i.e., Landing, Staging and DWH.

**8. Lookup Transformation**

Lookup Transformation is an SSIS component that performs lookups by joining data in input column with columns in the reference dataset.

It joins columns from input data set with columns in reference data set and we can additional columns from reference dataset for matching records.

**9.Derived Columns**

Derived columns Transformation is a SSIS component that can be used to modify the value of column before loading into target.

**For Example:** If we are loading a data from different sources and in that we have GENDER column and it value in Source1 is **‘FEMALE’ and ‘MALE’** and in source2 value is **‘F’ and ‘M’** then we want that value should be either converted to Female and male or F and M in target, that thing is possible by using Derived column transformation.

**Test Case Scenarios**

To test simple table loaded from source into target.

Test Scenarios:

Test Case Scenario can be identified by using logic called permutation and combination, and from test case scenario we can derive multiple test cases.

**Test Case Scenario**

1.Structure Validation (Meta Data Validation)

2.Count Validation

3.Duplicate Validation

4.Column Level Validation (Transformation validation)

5.Data Validation

**Test scenario defines the functionality and Test case defines navigational statement to define functionality.**

**Test scenario defines what to test?**

**Test case defines how to test?**

**Test Case Design**

While designing test case we need take care of below things

1.It should be simple.

2.Easy to understand

3.It should be cover all the functionality w.r.to requirement.

4.It should be specific or precise.

**NOTE:**

**Test cases always written in spreadsheet (excel- .xlsx) format.**

Whenever you are writing test cases then we need to follow test case template if it is existed or provided by you in team or we need to create it by our own based on requirement.

**Test Case Template**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case Name** | **Description** | **Test Step** | **Description (Design Steps)** | **Expected Result** |

**Testing Evidences**

Testing evidences are important after completion of test case execution, either test case is pass or fail, in both situation we must prepare evidence.

If test case pass, then we must attach evidence as execution proof to particular test case.

If test failed, then we must raise defect and attach the same evidence to highlight the issue.

**Sampling Approach:**

When source is having 10 lakh records and same has been loaded into the target then we need to adopt sampling approach to validate data from source to target with prior approval from stack holder (Client).

Selective records we will select and will validate from source to target.

**Q: If we have 1 lakh records in source and same has been loaded into target then what will be your approach to validate data?**

**ETL Project:**

Requirement:

We need to Prepare Order DWH.

**For XYZ** Organization And they need to maintain their WO (Work Order) Information at one place.

Data is stored in three different locations

**Source1** - .txt

**Source2** - .xlsx

**Source3** - .xlsx

**Design Statement:**

**Landing Layer:**

Load data from different source format and convert it into structural format at Landing Layer as per Source to Landing Mapping sheet.

Exact copy of source data but converted in Structural format.

**Staging Layer:**

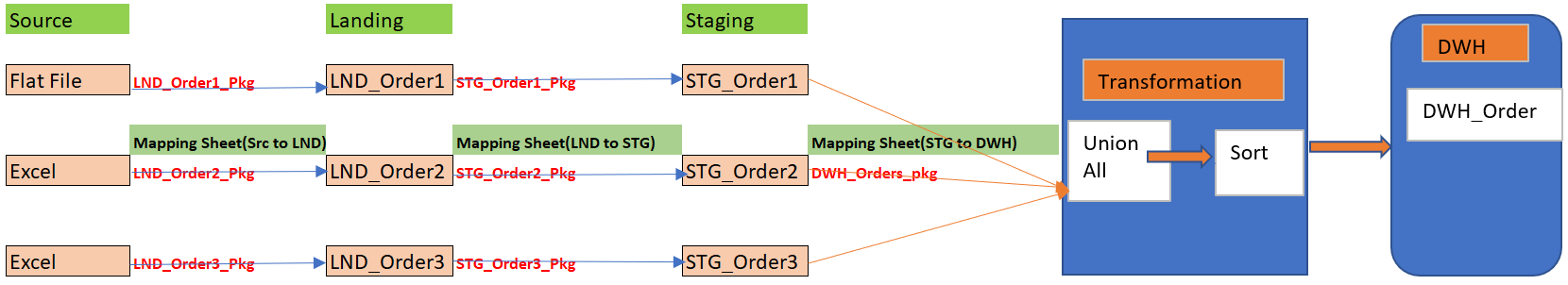
By applying transformation will Convert all the data from three landing tables into same format as per mapping sheet.

**DWH**

In this layer we combine all the Staging tables data and load into DWH layer as per mapping sheet.

**Documents**

Project flow diagram designed by Principal architect.



***Phase-I phase-II phase-III***

Documentation part will be design by BA and below are some of the documents

**SRS/BRS/FSD (Word Format)**

**Mapping sheet (Excel Format) w.r.t layer by layer**

Whenever documentation part is ready then it will circulate to both Developer and Tester and they will work parallelly.

|  |  |
| --- | --- |
| **Tester’s Role** | **Developers Role** |
| Identification of test scenarios | As per Mapping sheet he will create database and tables and then Package Creation. |
| Test case design | Unit Testing |
| Test case review with BA | Deploy the packages in test environment |
| Final Sign-off from BA. | He will circulate email to complete team which includes BA, DEV team and Test Team. |