# Aim:

To perform various OLAP operations such as slice, dice, drilldown,

rollup, pivot operations.

# Theory:

## ETL process:

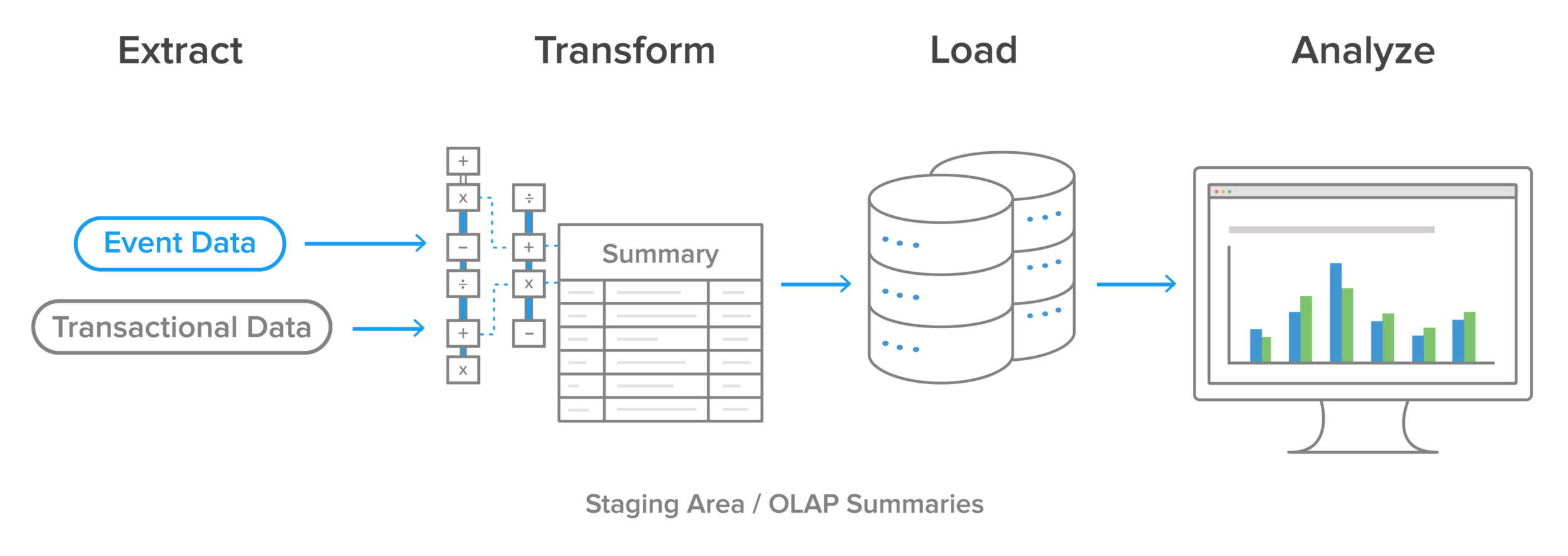
ETL is the process by which data is extracted from data sources (that are not optimized for analytics), and moved to a central host (which is). The exact steps in that process might differ from one ETL tool to the next, but the end result is the same.

At its most basic, the ETL process encompasses data extraction, transformation, and loading. While the abbreviation implies a neat, three-step process – extract, transform, load – this simple definition doesn’t capture:

* The transportation of data
* The overlap between each of these stages
* How new technologies are changing this flow

## **Traditional ETL process**

Historically, the ETL process has looked like this:



the ETL process: extract, transform and load. Then analyze.

## OLTP versus OLAP:

OLTP is operational, while OLAP is informational. A glance at the key features of both kinds of processing illustrates their fundamental differences, and how they work together.

|  |  |  |
| --- | --- | --- |
|  | OLTP | OLAP |
| Characteristics | Handles a large number of small transactions | Handles large volumes of data with complex queries |
| Query types | Simple standardized queries | Complex queries |
| Operations | Based on INSERT, UPDATE, DELETE commands | Based on SELECT commands to aggregate data for reporting |
| Response time | Milliseconds | Seconds, minutes, or hours depending on the amount of data to process |
| Design | Industry-specific, such as retail, manufacturing, or banking | Subject-specific, such as sales, inventory, or marketing |
| Source | Transactions | Aggregated data from transactions |
| Purpose | Control and run essential business operations in real time | Plan, solve problems, support decisions, discover hidden insights |
| Data updates | Short, fast updates initiated by user | Data periodically refreshed with scheduled, long-running batch jobs |
| Space requirements | Generally small if historical data is archived | Generally large due to aggregating large datasets |
| Backup and recovery | Regular backups required to ensure business continuity and meet legal and governance requirements | Lost data can be reloaded from OLTP database as needed in lieu of regular backups |
| Productivity | Increases productivity of end users | Increases productivity of business managers, data analysts, and executives |
| Data view | Lists day-to-day business transactions | Multi-dimensional view of enterprise data |
| User examples | Customer-facing personnel, clerks, online shoppers | Knowledge workers such as data analysts, business analysts, and executives |
| Database design | Normalized databases for efficiency | Denormalized databases for analysis |

## Different OLAP Queries:

There are five basic analytical operations that can be performed on an OLAP cube:

* Drill down: In drill-down operation, the less detailed data is converted into highly detailed data. It can be done by:
  1. Moving down in the concept hierarchy
  2. Adding a new dimension
  3. In the cube given in the overview section, the drill down operation is performed by moving down in the concept hierarchy of Time dimension (Quarter -> Month).
* Roll up: It is just opposite of the drill-down operation. It performs aggregation on the OLAP cube. It can be done by:
  + Climbing up in the concept hierarchy
  + Reducing the dimensions
  + In the cube given in the overview section, the roll-up operation is performed by climbing up in the concept hierarchy of Location dimension (City -> Country).
* Dice: It selects a sub-cube from the OLAP cube by selecting two or more dimensions. In the cube given in the overview section, a sub-cube is selected by selecting following dimensions with criteria:
  + Location = “Delhi” or “Kolkata”
  + Time = “Q1” or “Q2”
  + Item = “Car” or “Bus”
* Slice: It selects a single dimension from the OLAP cube which results in a new sub-cube creation. In the cube given in the overview section, Slice is performed on the dimension Time = “Q1”.
* Pivot: It is also known as rotation operation as it rotates the current view to get a new view of the representation. In the sub-cube obtained after the slice operation, performing pivot operation gives a new view of it.

# Implementation:

## Table Descriptions:

|  |
| --- |
| SQL> desc client;  Name Null? Type  ----------------------------------------- -------- ----------------------------  CLIENT\_ID NOT NULL VARCHAR2(5)  C\_NAME VARCHAR2(25)  ADDRESS VARCHAR2(50)  AGE NUMBER(3)  PHONE\_NO NUMBER(11)  GENDER VARCHAR2(2)  SQL> desc rooms;  Name Null? Type  ----------------------------------------- -------- ----------------------------  ROOM\_ID NOT NULL VARCHAR2(5)  ROOM\_TYPE VARCHAR2(15)  ROOM\_OCCUPANCY NUMBER(2)  PRICE NUMBER(6)  SQL> desc rdate;  Name Null? Type  ----------------------------------------- -------- ----------------------------  DATE\_ID NOT NULL VARCHAR2(5)  YEAR NUMBER(10)  MONTH VARCHAR2(10)  DAY VARCHAR2(10)  WEEK NUMBER(10)  DDATE DATE  SQL> desc city;  Name Null? Type  ----------------------------------------- -------- ----------------------------  CITY\_ID NOT NULL VARCHAR2(5)  COUNTRY VARCHAR2(20)  C\_NAME VARCHAR2(25)  HOTLIER\_ID VARCHAR2(5)  HOTLIER\_NAME VARCHAR2(20)  SQL> desc factt;  Name Null? Type  ----------------------------------------- -------- ----------------------------  CLIENT\_ID NOT NULL VARCHAR2(5)  CITY\_ID NOT NULL VARCHAR2(5)  DATE\_ID NOT NULL VARCHAR2(5)  ROOM\_ID NOT NULL VARCHAR2(5)  NO\_OF\_OVERNIGHT\_STAY NUMBER(3)  PROFIT NUMBER(8)  EXPENSE NUMBER(8) |

## Analytical Queries for system:

1. Total number of rooms booked by gender country wise with subtotal and grandtotal (Use rollup()).

|  |
| --- |
| SQL> select c.country,cl.gender,count(f.Room\_id) from city c,client cl,factt f where f.city\_id=c.city\_id and f.client\_id=cl.client\_id group by rollup(c.country,cl.gender);  COUNTRY GE COUNT(F.ROOM\_ID)  -------------------- -- ----------------  USA F 9  USA M 19  USA 28  India F 7  India M 12  India 19  Spain F 14  Spain M 8  Spain 22  Berlin F 3  Berlin M 7  Berlin 10  France F 6  France M 4  France 10  Russia F 2  Russia M 9  Russia 11  100  19 rows selected. |

1. How many reservations have been made in all countries till date.

|  |
| --- |
| SQL> select c.country,count(f.CITY\_Id) from factt f , city c where c.city\_id=f.city\_id group by (c.country);  COUNTRY COUNT(F.CITY\_ID)  -------------------- ----------------  Berlin 10  USA 28  Spain 22  France 10  India 19  Russia 11  6 rows selected |

1. Sum of expenses in all countries room and countrywise using cube().

|  |
| --- |
| SQL> select c.country,r.room\_type,sum(f.expense) from factt f, city c,rooms r where f.city\_id=c.city\_id and r.room\_id=f.room\_id group by cube (c.country,r.room\_type);  COUNTRY ROOM\_TYPE SUM(F.EXPENSE)  -------------------- --------------- --------------  1562000  Deluxe 53000  Luxury 28500  Sea View 54500  Club Room 94500  Penthouse 510000  Signature 30500  Luxury Grand 112500  Presidential 291500  Super Deluxe 255000  Super Luxury 132000  USA 494000  USA Deluxe 18000  USA Luxury 9000  USA Sea View 18000  USA Club Room 25000  USA Penthouse 160000  USA Signature 9500  USA Luxury Grand 2500  USA Presidential 131500  USA Super Deluxe 75000  USA Super Luxury 45500  India 179500  India Deluxe 12000  India Luxury 15500  India Sea View 20000  India Club Room 5000  India Penthouse 10000  India Signature 1500  India Presidential 85000  India Super Deluxe 25000  India Super Luxury 5500  Spain 357500  Spain Deluxe 10000  Spain Sea View 11000  Spain Club Room 31500  Spain Penthouse 130000  Spain Signature 10000  Spain Luxury Grand 60000  Spain Presidential 75000  Spain Super Luxury 30000  Berlin 199000  Berlin Sea View 1000  Berlin Club Room 33000  Berlin Penthouse 10000  Berlin Luxury Grand 15000  Berlin Super Deluxe 140000  France 138500  France Deluxe 11000  France Penthouse 60000  France Signature 1500  France Super Deluxe 15000  France Super Luxury 51000  Russia 193500  Russia Deluxe 2000  Russia Luxury 4000  Russia Sea View 4500  Russia Penthouse 140000  Russia Signature 8000  Russia Luxury Grand 35000  60 rows selected. |

1. Which city gives the maximum profit with respective rooms for all(Use Rank()).

|  |
| --- |
| SQL> select c.city\_id,f.room\_id,sum(f.profit),RANK() OVER (PARTITION BY f.room\_id ORDER by sum(f.profit)) AS Rank from factt f, city c where c.city\_id=f.city\_id group by (f.room\_id,f.profit,c.city\_id);  CITY\_ ROOM\_ SUM(F.PROFIT) RANK  ----- ----- ------------- ----------  C07 R01 15000 1  C03 R01 25000 2  C10 R01 25000 2  C01 R01 25000 2  C06 R01 35000 5  C01 R01 40000 6  C09 R01 40000 6  C04 R01 45000 8  C09 R01 45000 8  C02 R01 50000 10  C10 R01 50000 10  C04 R01 75000 12  C09 R01 75000 12  C10 R01 85000 14  C07 R01 95000 15  C05 R01 95000 15  C02 R01 120000 17  C05 R02 2000 1  C03 R02 20000 2  C02 R02 25000 3  C03 R02 25000 3  C08 R02 30000 5  C05 R02 50000 6  C08 R02 50000 6  C02 R02 90000 8  C01 R02 100000 9  C10 R03 10000 1  C08 R03 32000 2  C04 R03 40000 3  C06 R03 60000 4  C06 R03 64000 5  C10 R03 69000 6  C08 R04 10000 1  C01 R04 10000 1  C07 R04 15000 3  C04 R04 20000 4  C07 R04 75000 5  C04 R04 80000 6  C05 R04 80000 6  C08 R05 1000 1  C02 R05 2000 2  C06 R05 4000 3  C06 R05 4000 3  C06 R05 8000 5  C02 R05 22000 6  C05 R05 35000 7  C07 R06 2000 1  C02 R06 2000 1  C09 R06 2000 1  C05 R06 7500 4  C09 R06 22500 5  C06 R06 35000 6  C04 R07 1500 1  C10 R07 1500 1  C05 R07 1500 1  C07 R07 3000 4  C07 R07 3000 4  C04 R07 3500 6  C02 R07 8000 7  C01 R07 8000 7  C07 R07 8000 7  C04 R07 8000 7  C09 R07 8000 7  C08 R07 8000 7  C03 R07 8000 7  C10 R07 16000 14  C01 R07 32000 15  C03 R07 32000 15  C03 R08 2000 1  C08 R08 3000 2  C05 R08 4000 3  C08 R08 5000 4  C09 R08 5000 4  C05 R08 5000 4  C08 R08 6000 7  C10 R08 6000 7  C01 R09 1500 1  C02 R09 1500 1  C06 R09 1500 1  C05 R09 5000 4  C02 R09 5000 4  C09 R09 6500 6  C03 R09 10000 7  C04 R10 2000 1  C03 R10 2000 1  C01 R10 2000 1  C07 R10 4000 4  C09 R10 6000 5  C07 R10 6000 5  C10 R10 8000 7  90 rows selected. |

1. Which city gives the maximum profit with respective rooms (Use DENSE\_Rank()).

|  |
| --- |
| SQL> select c.city\_id,f.room\_id,sum(f.profit),DENSE\_RANK() OVER (PARTITION BY f.room\_id ORDER by sum(f.profit)) AS Rank from factt f, city c where c.city\_id=f.city\_id group by (f.room\_id,f.profit,c.city\_id);  CITY\_ ROOM\_ SUM(F.PROFIT) RANK  ----- ----- ------------- ----------  C07 R01 15000 1  C03 R01 25000 2  C10 R01 25000 2  C01 R01 25000 2  C06 R01 35000 3  C01 R01 40000 4  C09 R01 40000 4  C04 R01 45000 5  C09 R01 45000 5  C02 R01 50000 6  C10 R01 50000 6  C04 R01 75000 7  C09 R01 75000 7  C10 R01 85000 8  C07 R01 95000 9  C05 R01 95000 9  C02 R01 120000 10  C05 R02 2000 1  C03 R02 20000 2  C02 R02 25000 3  C03 R02 25000 3  C08 R02 30000 4  C05 R02 50000 5  C08 R02 50000 5  C02 R02 90000 6  C01 R02 100000 7  C10 R03 10000 1  C08 R03 32000 2  C04 R03 40000 3  C06 R03 60000 4  C06 R03 64000 5  C10 R03 69000 6  C08 R04 10000 1  C01 R04 10000 1  C07 R04 15000 2  C04 R04 20000 3  C07 R04 75000 4  C04 R04 80000 5  C05 R04 80000 5  C08 R05 1000 1  C02 R05 2000 2  C06 R05 4000 3  C06 R05 4000 3  C06 R05 8000 4  C02 R05 22000 5  C05 R05 35000 6  C07 R06 2000 1  C02 R06 2000 1  C09 R06 2000 1  C05 R06 7500 2  C09 R06 22500 3  C06 R06 35000 4  C04 R07 1500 1  C10 R07 1500 1  C05 R07 1500 1  C07 R07 3000 2  C07 R07 3000 2  C04 R07 3500 3  C02 R07 8000 4  C01 R07 8000 4  C07 R07 8000 4  C04 R07 8000 4  C09 R07 8000 4  C08 R07 8000 4  C03 R07 8000 4  C10 R07 16000 5  C01 R07 32000 6  C03 R07 32000 6  C03 R08 2000 1  C08 R08 3000 2  C05 R08 4000 3  C08 R08 5000 4  C09 R08 5000 4  C05 R08 5000 4  C08 R08 6000 5  C10 R08 6000 5  C01 R09 1500 1  C02 R09 1500 1  C06 R09 1500 1  C05 R09 5000 2  C02 R09 5000 2  C09 R09 6500 3  C03 R09 10000 4  C04 R10 2000 1  C03 R10 2000 1  C01 R10 2000 1  C07 R10 4000 2  C09 R10 6000 3  C07 R10 6000 3  C10 R10 8000 4  90 rows selected. |

1. What is the overall profit from 13-May-12 to 04-Nov-15 per city ?(SLICE)

|  |
| --- |
| SQL> select c.city\_id,sum(f.profit) from factt f,rdate r,city c where (r.ddate >= '13-May-12' and r.ddate <='04-NOV-15') and r.date\_id=f.date\_id and c.city\_id=f.city\_id group by(c.city\_id) order by c.city\_id asc;  CITY\_ SUM(F.PROFIT)  ----- -------------  C01 169000  C02 90000  C03 82000  C04 15000  C05 159000  C06 34000  C07 97000  C08 40000  C09 61500  C10 119500  10 rows selected. |

1. What is the overall profit and expense from 15-Oct-15 to 01-Jul-19 ?(DICE)

|  |
| --- |
| SQL> select c.city\_id,ro.room\_id,sum(f.expense) from factt f,rdate r,city c,rooms ro where (r.ddate >= '15-Oct-15' and r.ddate <='01-Jul-19') and r.date\_id=f.date\_id and c.city\_id=f.city\_id and c.city\_id ='C02'and ro.room\_id=f.room\_id group by(c.city\_id,ro.room\_id);  CITY\_ ROOM\_ SUM(F.EXPENSE)  ----- ----- --------------  C02 R06 30000  C02 R01 20000  C02 R05 31500  C02 R02 75000 |

1. What is the total number of overnight stays overall in all hotels.

|  |
| --- |
| SQL> select sum(no\_of\_overnight\_stay) as Total\_stay\_days from factt;  TOTAL\_STAY\_DAYS  ---------------  330 |

1. Total number of rooms allocated of all locations in India.

|  |
| --- |
| SQL> select count(room\_id) from factt where CITY\_ID='C03'or CITY\_ID='C08';  COUNT(ROOM\_ID)  --------------  19 |

1. What are the total Expenses of a Spanish Branches ?

|  |
| --- |
| SQL> select sum(expense) from factt where CITY\_ID='C07' or CITY\_ID='C02';  SUM(EXPENSE)  ------------  357500 |

# Conclusion:

In this experiment, we understood what OLAP is Online analytical Processing and how it differs from OLTP that is online transaction processing. We then learnt the operations we can perform in OLAP. We implemented all operations like roll up, drill down, rank, dense rank on the problem statement of the hotel management system.