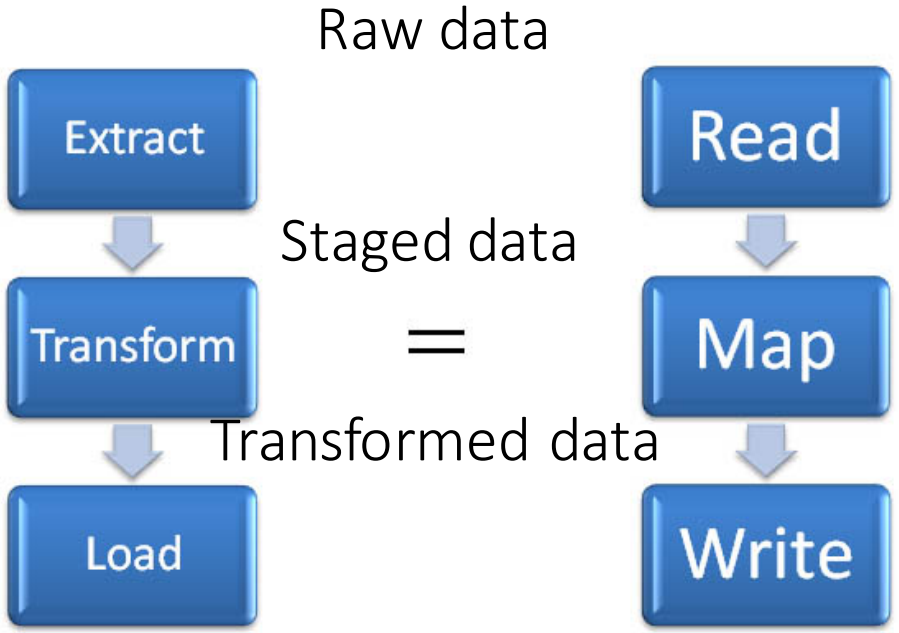
Term Project – Supplementary Slides

Scenario

1. TPS-type data available to database analysts and data scientists
   - Examples: List of employees, their joining dates, salaries, training given, department responsibilities, locations, bosses, performance evaluations
   - Web searches on Expedia – date/time, details of search, outcome, username, location
2. Analysts understands this data in detail and models it
3. Analysts merges this data with other data (SQL)
   - create initial queries to check quality of data, answer business questions
   - Data is still normalized and TPS-like
4. Analysts interview users of the reporting applications needed by executives
5. Analysts write SQL code to set up the data-warehouse
   - And populate it with fresh data regularly

Part A

- Finalize the data model (ERD)
- Create the empty database (CREATE TABLE, ALTER TABLE)
- Populate the database
  - **Extract**: Identify the raw data sources and extract the data into the 'staging' area (SELECT, CREATE VIEW, LOAD DATA INFILE)
  - **Transform**: Match data types, clean/insert NULLs (UPDATE, ALTER TABLE)
  - **Load**: Populate the empty database (LOAD DATA INFILE, INSERT/UPDATE)



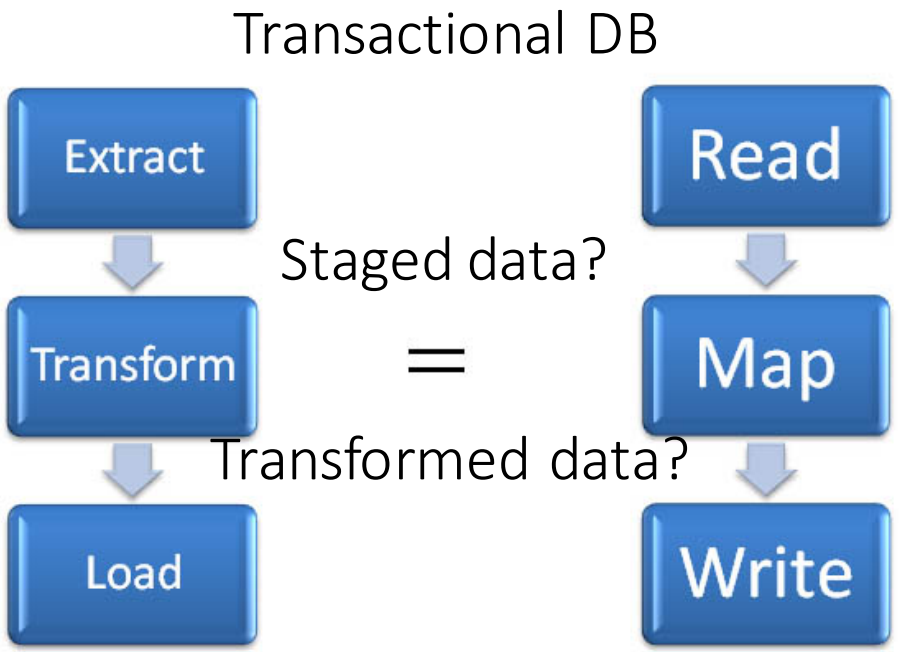
----Part A over---

..cont'd

- For the ETL, the input data is that which is:
  - Provided by me
  - Downloaded by you from the Internet (e.g., zip codes, airport destination, etc.)
  - Popular databases (see SimplyMap in the SFU library)
  - Generated by you using the Random function within MySQL/Excel
- The outcome of this ETL process is the Transactional DB corresponding to the ERD

Part B: Apply the same ETL to Transactional DB

- Decide on the strategic/research questions that can be asked for executive decision-making by creating a Data Mart from the Transaction DB
- Apply the ETL process as before

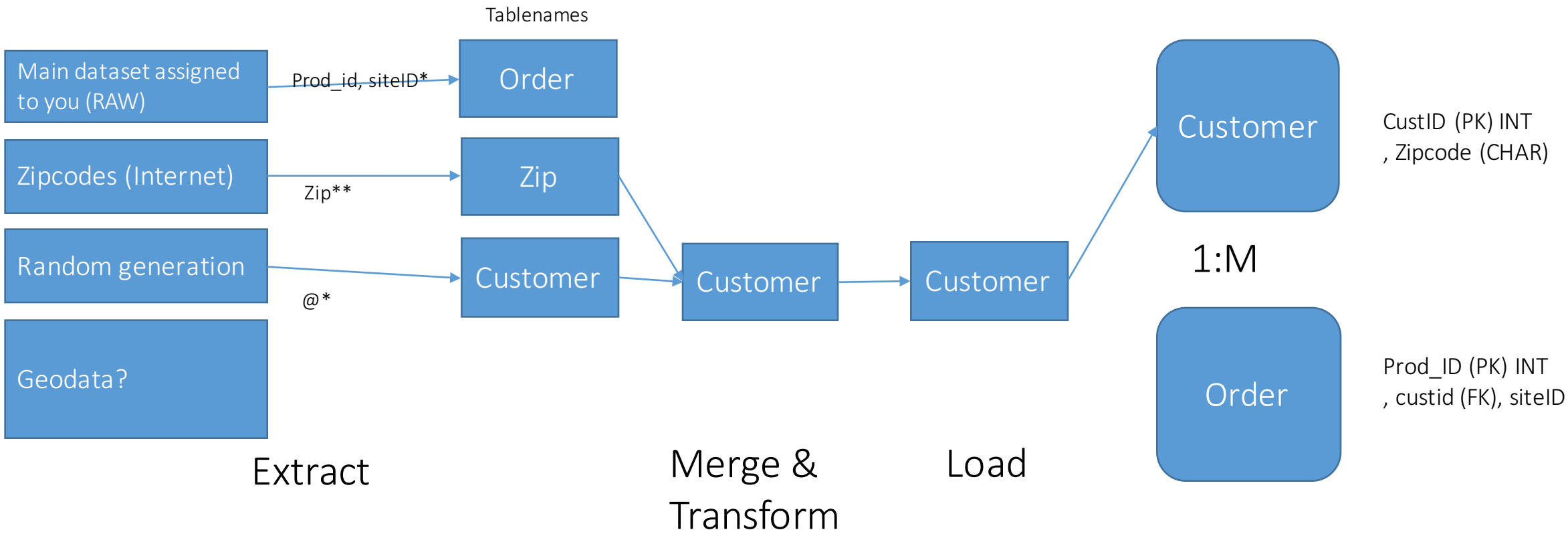


----Part B over---

Cont'd

- For ETL in Part B, the input data is the transactional DB you will have at the end of Part A
- The outcome of this ETL process is the (Analytical DB/Data Mart) that answers the strategic questions you have identified.

Illustrating the ETL for Part A



ETL Demo – Ensuring referential integrity when merging customer data

- **When all given data is real,** you would be given CustomerIDs and you have to make sure that every CustomerID in Product Table exists in Customer Table. If not, the data is not clean because you cannot trace who bought a particular product for which you do not have a corresponding record in Customer Table.
  - If so, MySQL won't let you define Customer ID as a FK in Search when creating the Product table.
- **For generating some of the raw data,** ensure this is the case by
  - Finding all unique order_ids from data assuming every order is placed by a single Customer (see ERD)

    create table tmpOrder as select distinct(orderID, date) from raw;
  - Then randomly generating CustomerID for inserting into Order

    create table tmpOrder_1 as select orderId, (FLOOR(1 + RAND() * (20000 - 0))) as custID from tmpOrder;
  - Then loading all unique CustomerIDs from Order into Customer table

    create table customer_1 as select distinct(custid) from tmpOrder_1;

    ALTER TABLE customer ADD PRIMARY KEY (`custid`);
  - Outcome: Each Order in Order table has a CustID and all such CustIDs are present in the Customer table

ETL Demo – Ensuring referential integrity when importing Zip code (or any text data)

- Adding Zipcode as an additional column to Customer_1

```
ALTER TABLE customer_1 modify COLUMN zipcode VARCHAR(5) NULL AFTER custid;
```

- Prepare another table to receive raw data from .csv file. Note the PK here.

```
create table zip(zipid int auto_increment primary key, zipcode char(5));
LOAD DATA LOCAL INFILE 'C:/Temp/ziptmp.txt' INTO TABLE zip fields terminated by ',' enclosed by '"' lines terminated by '\n'(`zipcode`);
```

- Note how the PK gets filled up automatically and serially --- we will use this key to join Zip and Customer_1
- Now we link each zipcode randomly to various customers in Customer_1 Table. Since there are 42741 zip codes , add a column to customer and assign it a random number 1, 42741

```
create table Customer_2 as select custid, (FLOOR(1 + RAND() * (20000 - 0) )) as zipID from customer_1;
```

- Customer_2 and Zip – both have zipid as a column. We use a join, keep zipcode (and NOT zipid)

```
create table customer_3 as select custid, zipcode from customer_2 c, zip z where c.zipid = z.zipid;
```

- EXTRACT and MERGE is complete. Now TRANSFORM the CustID to a FK. Then LOAD as follows:

```
drop table customer_2, zip;    ALTER TABLE customer_3 RENAME TO customer;
```