Introduction to Database & BigQuery Setup

Problem Statement

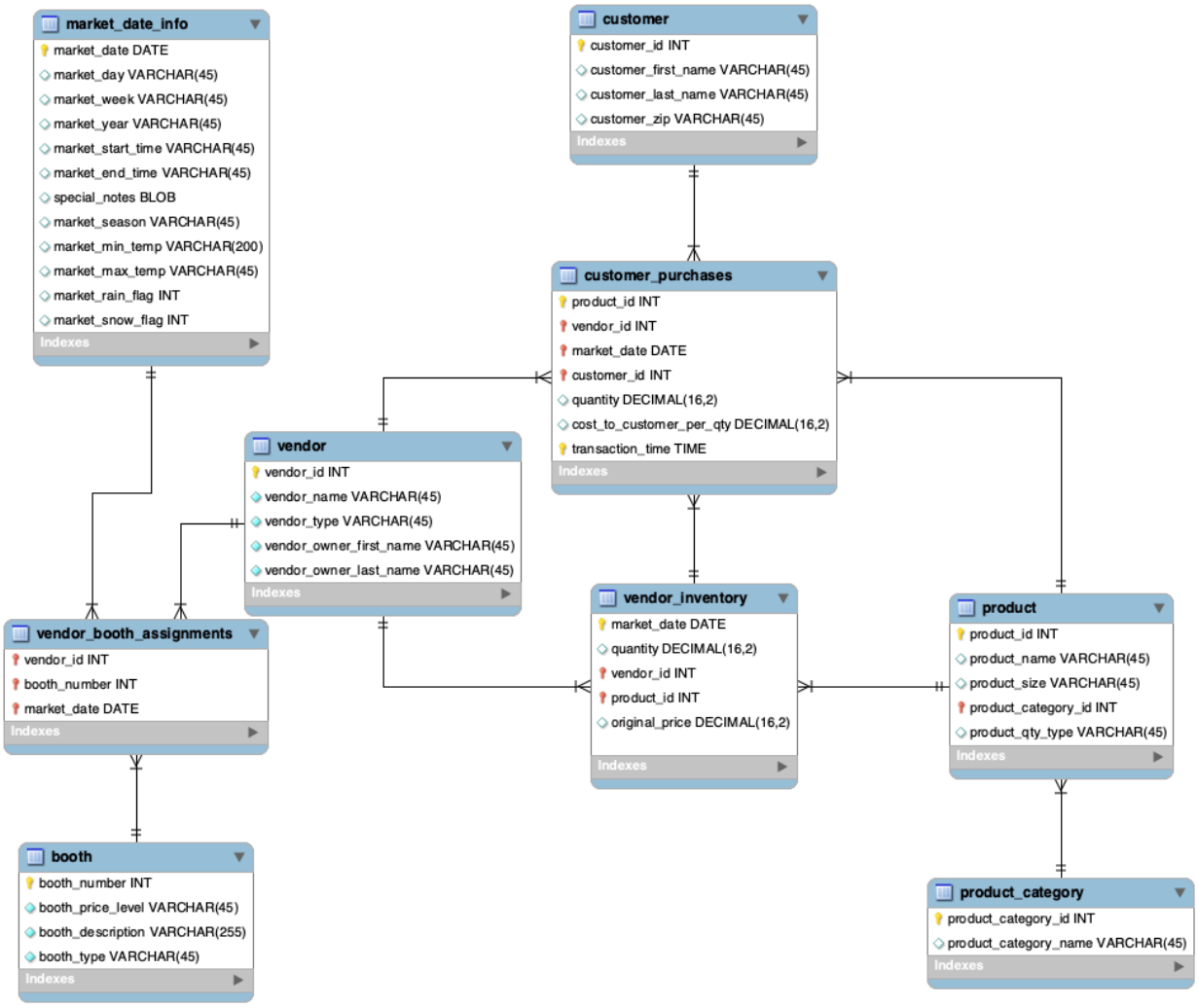
- It's your first day as a Data Analyst at **Amazon Fresh.**

- Amazon Fresh has recently decided to open big stores where farmors can directly sell their produce from their stores.

- **All the data is stored in a Database.**

- Your manager reaches out to you and gives you access to a **DBMS and a DB schema** that looks like this.

- He tells you that we need to grow our business (increase the revenue generated) through these stores and asks you to derive all the customer and product-related insights from this schema?



First thoughts come to your mind:

- Where is this data stored? DB?
- What is a DBMS?
- What is a DB schema?
- What does each box represent in this diagram?
- What do those lines represent?

Where is this data stored? - DBMS

Your manager tells you that the data is stored in a Relational DBMS called MySQL.

-> What is a DBMS?

- Any interaction you make with an app, **e.g., Amazon**, searching from the list of products and categories, wish-listing a product, adding to a cart, and placing an order.
- Whatever web pages we visit or applications we use, all of them use some backend system to collect all the data.
- **All of these backend systems are connected to Databases** that store all the interactions we make with their app.

DBMS consists of two phrases:

1. **Database** - a collection of interrelated tables

2. **Management System**
    a. A set of operations that help in managing & manipulating the Database.
    b. Example of the operations -
        i. CRUD - Create, Read, Update, Delete
        ii. Search, Insert, Grant, Revoke, etc.

In simple words, a **DBMS consists of related data and a set of programs to access & manipulate that data.**

→ Now, the tables in which we store data are also called **relations.**

→ And thus, this type of DBMS, which stores data in a **tabular** format, is called **Relational DBMS** or **RDBMS.**

→ Examples of **RDBMS** include:

- Oracle
- MySQL
- SQLite
- Microsoft SQL Server
- PostgreSQL

→ Then, there are **Non-relational DBMS** too, but we'll talk about them later.

Question: You should be wondering, why do we even need a **Database** if we already have **Excel**?

1. **Scalability**:
    - Databases can handle large datasets and scale effectively as data size increases.
    - Whereas Excel has limitations on the number of rows.
2. **Performance**:
    - Databases are optimized for querying and retrieving data quickly.
    - In contrast, Excel can become very slow when performing calculations or filtering large amounts of data.
3. **Data Integrity**:
    - Databases enforce certain conditions (also known as constraints) ensuring that the data is accurate.
    - Excel, on the other hand, is more prone to human errors.
4. **Concurrent Access**:
    - Databases allow multiple users to access and modify data simultaneously.
    - Excel, on the other hand, can lead to inconsistency in data due to multiple user access.
5. **Security**:

    - Databases allow administrators to manage user access & modification rights.
    - Excel files, on the other hand, can be easily shared and copied.

Now that the purpose and meaning of DBMS are clear, the next question is **how to figure out what all tables, columns, and rows are there in a relational database?**

That's where DB schema comes into the picture:

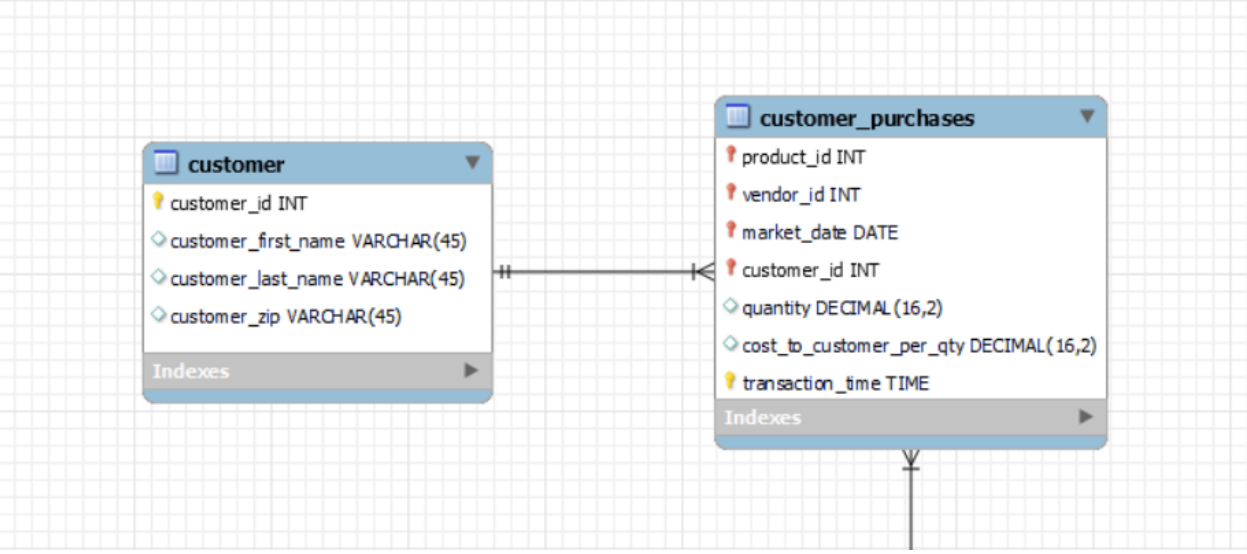
What is a DB schema or an Entity-Relationship Diagram?

- A database schema represents how the data is organized and provides information about the relationships between the tables in a given database.

- In our DB schema, those boxes that you see represent individual **tables/relations**.

- The lines connecting those tables with each other are called **relationships**. We'll talk about different kinds of relationships later at some point in this module.

- Notice that along with the **column/attribute** names, a bit of extra detail is also provided. This is called the **data type** of a column.



- It depicts that for each column, what type of data is stored in that column.

There are three main data types that we need to know about:

1. **string,**
2. **numeric, and**
3. **date and time.**

String → Char(), Varchar(), etc.

- Char is **Fixed** Length Character String
  - E.g. for Char(3) data type, a valid data entry could be IND, AUS, USA
  - Whereas inputs like INDIA, IN will throw an error.
  - Because it'll require exactly '3' characters.
- Varchar is **Variable** Length Character String
  - E.g. for Varchar(5) data type, a valid entry could be IN, IND, INDIA
  - Whereas inputs like AMERICA, AUSTRALIA will throw an error.
  - Because it'll accept only up to '5' characters.

Numeric → Int64, Float64, etc.

- Int64 is for storing an **integer value.**
- Float64 is for storing a **floating-point value.**

Date & Time → Date, Time, Datetime, Timestamp

- Date data type is used for storing a date in the format YYYY-MM-DD.
    - E.g., 2023-05-17
- Time data type is used for storing a time in the format hh:mm:ss.
    - E.g., 03:07:00
- Datetime is used for storing a combination of date & time in the format YYYY-MM-DD hh:mm:ss.
    - E.g., 2023-05-17 03:07:00
- Timestamp values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC).

NOTE:

- **In BigQuery,** the following documentation page can help you with these: Data Types
- **In MySQL**, you can refer to this link to showcase different data types. https://www.w3schools.com/sql/sql_datatypes.asp

---

Concept of Keys

Why do we need a key?

Keys in relational databases help to specify constraints on the data in a table.

Let's understand the essence of different keys using the `**customers**` table given below.

customer_id	name	phone	age
1	Akon	9876723452	17
2	Akon	9991165674	19
3	Bkon	7898756543	18
4	Ckon	8987867898	19
5	Dkon	9990080080	

Question: Which of these columns would you prefer to uniquely identify each record in the table?

Let's rule out which ones we can't use -

1. Name?
    a. No, because customers 1 & 2 have the same names.
2. Age?
    a. No, because customers 2 & 4 are of the same age.
    b. And we don't have the age for customer 5.

That means the column **must always be unique** and **can never be NULL**.

What do we call such columns?

Primary Key

- A primary key is a column or a set of columns that uniquely identifies each row in the table.

- Every table in a database should have a primary key because it ensures the uniqueness and integrity of the data.
- The primary key's value must be assigned when inserting a record and can never be updated.
- Note that a relation is allowed to have only one primary key.

Question - In our example of `customer` table, among the 2 columns - customer_id and phone, which one should be the primary key and why?
Answer - The most appropriate column that should be the primary key is customer_id.

Question - But, why not a phone, even if it satisfies all the criteria? i.e., it's both unique and non-null.

Answer - It can be, but what if a customer wants to update his/her phone number!
In that case, it would require updating all related records in other tables that reference it.

Why?
Let us understand.

customer_id	aadhaar_number
1	4738 2956 7103
2	9123 7456 8521
3	1357 8964 2053
4	8765 4321 0987
5	

- The Aadhaar Card number is **Unique** for all the citizens of India.

- If it gets lost and another duplicate copy is issued, then the duplicate copy always has the same number as before. Thus, it is **non-updateable**.
- But since Aadhaar Card hasn't been made mandatory by the govt, some of the citizens might not have one.
- So, for them, its value would be **NULL**.

Unique Key

A unique key is:

- Unique for all the records of the table.
- Non-updatable, meaning its value cannot be changed once assigned.
- But may have a NULL value.

Foreign Key

Let us look at the `**orders**` table given below.

order_id	item	quantity	cust_id
1	Fruits	2.5	1
2	Veggies	3	1
3	Meat	1	3
4	Veggies	5	5
5	Dairy	2	2

Remember our `**customers**` table?

customer_id	name	phone	age
1	Akon	9876723452	17
2	Akon	9991165674	19

3	Bkon	7898756543	18
4	Ckon	8987867898	19
5	Dkon	9990080080	

Here,

- The customer_id / cust_id column is a common link between the `**orders**` and `**customers**` table.
- This means that the cust_id column in the `orders` table is used to establish a relationship with the corresponding customer_id in the `customers` table.
- So, if we want to fetch the order details of a customer with customer_id (say 3), we can do that by using the cust_id as a foreign key, referencing the primary key (customer_id) of the `customers` table.

Therefore,

- A foreign key is a column or a set of columns in a table that refers to the primary key of another table.
- It is used to establish a relationship between two tables by enforcing referential integrity.
- The values in the foreign key column(s) must match values in the primary key column(s) of the referenced table.
- A foreign key may have a name other than that of a primary key.

Candidate Key

Suppose elections are being conducted in your city for the mayor's position.

You have two different candidates out of which you have to choose any one whom you'll give your vote to.

What you have to notice here is that although you have two available candidates, only one of them can be the city mayor at a time.

The same thing is with candidate keys.

- Any column or set of columns that can act as a primary key is referred to as a candidate key.
- Every table must have at least a single candidate key (obviously can have multiple candidate keys) which may have single or multiple attributes.
- In our example, customer_id and phone both are candidate keys for the `customers` table.
- **Note** that candidate keys are just a theoretical concept. They're not used in real world applications.

Typical day of a Data Scientist/Analyst working in the industry

- In any company, data engineers set up a data warehouse. They collect the data from different sources and dump it all into this warehouse in a structured manner.
- Data analyst/scientist/ML engineer needs the historical data from the warehouse in order to build their model and perform the analytics on top of it.
- A data analyst/scientist would get the data from the warehouse using SQL queries. It is preprocessed and sanitized. Finally using it for their purpose.

What is the difference between a Database vs Data Warehouse?

- Data warehouse is a giant database optimized for analytics use cases.

- A data warehouse is called an **OLAP** system whereas a DBMS is called an **OLTP** system.
- Companies choose cloud service providers for the data warehousing services.
- There are many cloud service providers like AWS, GCP, Azure, etc. that provide these data warehousing services.

The tool that we are going to work with is BigQuery over Google Cloud Platform (GCP).

Introduction to BigQuery -

- How to create a new project?
- How to create a new dataset?
- How to upload tables in a dataset?

The following document would help you in setting up the **`farmers_market`** database on BigQuery.
How_to_Set_up_BigQuery.pdf

Bigquery setup video: link
