**Datasets :**



**Datasets Explanation :**

* **order\_id:** A unique identifier for each order, automatically incremented.
* **customer\_name:** Name of the customer placing the order.
* **restaurant\_name**: Name of the restaurant from which the order is made.
* **order\_date:** The date and time when the order was placed.
* **delivery\_time:** The date and time when the order was delivered (or expected to be delivered).
* **delivery\_address**: The address to which the order is delivered.
* **city:** The city in which the order is delivered.
* **delivery\_status:** The current status of the order (e.g., 'Delivered', 'Cancelled', 'Pending').
* **order\_amount:** The total amount for the order.
* **delivery\_agent:** The name of the delivery agent handling the order.

**DDL :** Create the associated Table structure using appropriate datatype and with right timezone(**Asia/Kolkata here since Swiggy Operates in India**) and load the data both in **Snowflake** & **MySQL**

**Hint :** Focus on the **PK** column and think of generating a sequence 1,2,3…(do research)

**Problem Statement :**

**Easy Level:**

1. **Extracting Date Components:**
   * Extract the year, month, and day from the **order\_date** column in the Swiggy dataset.
2. **Current Timestamp:**

* Get the current timestamp and compare it with the delivery\_time.

1. **Date & Time Difference:**

* Calculate the number of days,hours,minutes,etc between the order\_date and delivery\_time and store it in respective columns.
* Add 45 minutes to the delivery\_time and show the updated time.

4. **Orders Placed in Specific Months:**

* Find all orders placed in September of any year.

**Intermediate Level:**

1. **Time Zone Conversion:**
   * Convert the delivery\_time from UTC to a specific time zone (e.g., 'Asia/Kolkata').
2. **Orders on Specific Weekends:**

* Find all orders placed on a weekend (Saturday or Sunday).

**Advanced Level:**

1. **Calculating Peak Hours:**
   * Identify the peak delivery hours by extracting the hour from delivery\_time and grouping by hour
   * Identify which day of the week has the most deliveries.
2. **Handling Daylight Saving Time:**
   * Convert the delivery\_time into a time zone that observes daylight saving time (e.g., 'America/New\_York') and check if any orders fall during the daylight saving adjustment period.
3. **Identify Late Deliveries:**

* Find orders where the delivery took more than 1 hour.

1. **Filtering Orders Between Two Date-Times:**
   * Find all orders placed between specific date ranges, e.g., between '2023-09-01' and '2023-09-05' and orders placed between 5 PM and 7 PM both for those dates included and without those date too irrespective of dates.
2. **Handling Leap Years:**
   * Find orders placed on February 29th (during leap years).
3. **Timestamp Arithmetic with Time Zones:**
   * Calculate the time difference between the order time in 'Asia/Kolkata' and 'America/Los\_Angeles'.
4. **Finding the Most Recent Order:**
   * Retrieve the most recent order placed in the last 7 days.

**Expert Level:**

1. **Calculate Average Delivery Time per City:**

* Calculate the average delivery time for each city.

1. **Finding Busiest Days by City:**

* Identify which day of the week has the highest number of orders for each city.

1. **Delayed Deliveries Based on Peak Hours:**

* Identify orders that took longer during peak hours (5 PM - 8 PM).

1. **Orders with Week-to-Week Growth:**

* Calculate week-on-week growth of orders.

1. **Finding Orders Affected by Public Holidays:**

* Identify orders placed on specific public holidays (e.g., New Year's Day, Diwali).