# SQL ASSIGNMENT

BY: ANJANA JOSHY (22022447)

## TOPIC: MOBILE SALE DASHBOARD

This assignment is about a mobile sale dashboard that displays customer and mobile details through tables: Customer, Mobile, Mobile sale and stock. I created a database using Python and, used Faker to make the data seem real. I had used some data types, here names is nominal, ratings is ordinal, production date is interval, and prices is ratio.

```
[3] !pip install Faker
    import pandas as pd
    import numpy as np
    from faker import Faker
    Fake = Faker()
```

```
[2] #setting 1000 rows
    n = 1000
    #customer : nominal
    customer_id = np.random.choice(np.arange(100,99999),size = n, replace = False)
    #CUSTOMER NAME : nominal
    customer_names = [Fake.name() for i in range(n)]
    #Phone number : nominal
    country_codes = np.array([f"+{Fake.random_int(1, 99)}" for _ in range(1000)])
    phone_number = np.array([f"{code} {Fake.random_number(10)}" for code in country_codes])

    # Create DataFrame
    df_customer = pd.DataFrame({
        'CustomerID': customer_id,
        'CustomerName': customer_names,
        'Phone Number': phone_number
        })

    df_customer.set_index('CustomerID', inplace=True)
    #SORT WITH CUSTOMER ID
    df_customer.sort_index(inplace = True)
    #saving as csv file
    df_customer.to_csv('customer.csv')
```

This code creates 1000 rows of fake information for each customer, including a unique CustomerID, Customer Name, and Phone Number. Following dataframe sorting by CustomerID, the resultant file is saved as a CSV file called “customer.csv”. The output of the dataframe is shown below.

| CustomerID | CustomerName | Phone Number |
|---|---|---|
| 101 | Laura Hall | +49 6091584074 |
| 155 | Edward Russell | +34 2669265862 |
| 433 | Donna Brown | +50 843041239 |
| 438 | Casey Harvey | +99 3434543663 |
| 553 | Brandon Sullivan | +33 6780378570 |
| ... | ... | ... |
| 99225 | Yvonne Turner | +84 9165802264 |
| 99337 | Matthew Payne | +18 8586076830 |
| 99623 | Katie Johnson | +67 5744120708 |
| 99874 | Jose Hood | +60 3237927609 |
| 99884 | Matthew Rivers | +21 3442356204 |

1000 rows × 2 columns

```
[ ] #mobile id : nominal
    mobile_id = np.random.choice(np.arange(100,9999),size = n,replace = False)
    # List of specific mobile models :nominal
    mobile_models = ['Galaxy S21', 'Galaxy Note 20', 'Galaxy A52', 'Galaxy Z Fold 3', 'Galaxy M32', 'Galaxy XCover Pro', 'Galaxy Z Flip 3', 'Galaxy A32', 'Galaxy S20', 'Galaxy A71']
    # Repeat the models to create 1000 rows
    repeated_models = np.random.choice(mobile_models, size=1000)

    #colors : nominal
    colors = ['Bora Purple','Phantom White','Green','Pink Gold','Phantom Black']
    # Repeat the models to create 1000 rows
    phone_color= np.random.choice(colors, size=1000)

    #producton date : interval
    production_year = np.random.randint(2000,2023,n)
    production_month = np.random.randint(1,13,n)
    production_day = np.random.randint(1,29,n)
    production_date =[f'{production_year[i]}-{str(production_month[i]).zfill(2)}-'f'{str(production_day[i]).zfill(2)}' for i in range(n)]
    # storage : ordinal
    #selecting storages
    storage = ['64','128','256','512' ]
    # Repeat the models to create 1000 rows
    storage= np.random.choice(storage, size=1000)

    #setting dataframe for storage
    df_mobile = pd.DataFrame({
        'storage':storage
        })

    random_addition = np.random.uniform(1, 2, n)
    #price : ratio
    price = df_mobile['storage'].astype(int) * 3 + random_addition
```

```
    #dataframe for table mobile
    df_mobile = pd.DataFrame({
        'mobileID': mobile_id,
        'models': repeated_models,
        'color': phone_color,
        'production date': production_date,
        'storage':storage,
        'price': price
        })

    df_mobile['price'] = df_mobile['price'].round(2)
    df_mobile.set_index('mobileID', inplace=True)

    #saving as csv file
    df_mobile.to_csv('mobile.csv')
```

This code generates a dataset of mobile, which contains the columns models, colors, production dates, storage capacities, and unique mobile IDs. Each mobile is assigned a model, color, and ID at random. The date of production is generated at random using the format “YYYY-MM-DD”. Random choices are made regarding the storage options (64GB, 128GB, 256GB, and 512GB), and prices are calculated based on storage with some random variations. After that, the data is saved as a CSV file called “mobile.csv” and the result is shown below.

| mobileID | models | color | production date | storage | price |
|---|---|---|---|---|---|
| 9893 | Galaxy A71 | Pink Gold | 2015-04-21 | 64 | 193.81 |
| 7748 | Galaxy A52 | Phantom Black | 2013-08-17 | 512 | 1537.56 |
| 2884 | Galaxy XCover Pro | Phantom White | 2004-02-20 | 512 | 1537.10 |
| 8519 | Galaxy A52 | Phantom Black | 2012-02-03 | 256 | 769.63 |
| 2460 | Galaxy M32 | Green | 2021-12-28 | 512 | 1537.45 |
| ... | ... | ... | ... | ... | ... |
| 6630 | Galaxy Z Flip 3 | Bora Purple | 2017-10-27 | 512 | 1537.25 |
| 820 | Galaxy S21 | Green | 2000-11-26 | 512 | 1537.83 |
| 1704 | Galaxy Note 20 | Bora Purple | 2007-09-20 | 256 | 769.48 |
| 5414 | Galaxy S20 | Bora Purple | 2011-09-15 | 512 | 1537.02 |
| 9327 | Galaxy M32 | Phantom Black | 2005-04-05 | 256 | 769.69 |

1000 rows × 5 columns

```
[13] #SETTING Stock ID: nominal
     stockid = np.random.choice(np.arange(100,9999),size = n, replace = False)
     #quantity avaialble: ratio
     quantity_available = np.random.randint(0,999,size = n)
     #dataframe for table stock
     df_stock = pd.DataFrame({
         'stock id': stockid,
         'quantity available':quantity_available,
         'models': repeated_models,
             })
     df_stock.set_index('stock id', inplace=True)
     #saving as csv file
     df_stock.to_csv('stock.csv')
     df_stock
```

| stock id | quantity available | models |
|---|---|---|
| 4807 | 218 | Galaxy A71 |
| 6010 | 62 | Galaxy A52 |
| 1631 | 915 | Galaxy XCover Pro |
| 9049 | 841 | Galaxy A52 |
| 4925 | 2 | Galaxy M32 |
| ... | ... | ... |
| 7024 | 337 | Galaxy Z Flip 3 |
| 9280 | 658 | Galaxy S21 |

This cod generates a dataset that represents stock items. To ensure uniqueness, each stock item is given a unique stock ID, which is generated at random. To represent the amount of each item in stock, random numbers between 0 and 999 are entered into the 'Quantity Available' column. Following, the data is placed into a dataframe called "df_stock", with the stock ID acting as the index. 'stock.csv' is the name of a CSV file that contains this dataframe and the output is shown above.

```
[14] #order id: nominal
     order_id = np.random.choice(np.arange(100,9999),size = n,replace = False)

     #sales channel:nominal
     sales_channels = np.random.choice(['Offline', 'Online'], size=n)
     # Generate random ratings:ordinal
     rating = np.random.randint(2, 11, size=n).astype(float)
     # Set some values to NaN
     n_null = 150
     random_indices = np.random.choice(rating.size, n_null, replace=True).astype(int)
     rating[random_indices] = np.nan

     #setting dataframe for table Mobilesale
     df_Mobilesale = pd.DataFrame({
         'mobileID': mobile_id,
         'order id': order_id,
         'sales channels': sales_channels,
         'rating': rating,
         'customer id': customer_id
             })
     # Replace NaN with a string representation
     df_Mobilesale['rating'] = df_Mobilesale['rating'].fillna('NaN')


     df_Mobilesale['CompoundKey'] = df_Mobilesale['order id'].astype(str) + '_' + df_Mobilesale['mobileID'].astype(str)
     df_Mobilesale.set_index('order id', inplace=True)
     df_Mobilesale.sort_index(inplace = True)

     #saving as csv file
     df_Mobilesale.to_csv('mobilesale.csv')
     df_Mobilesale
```

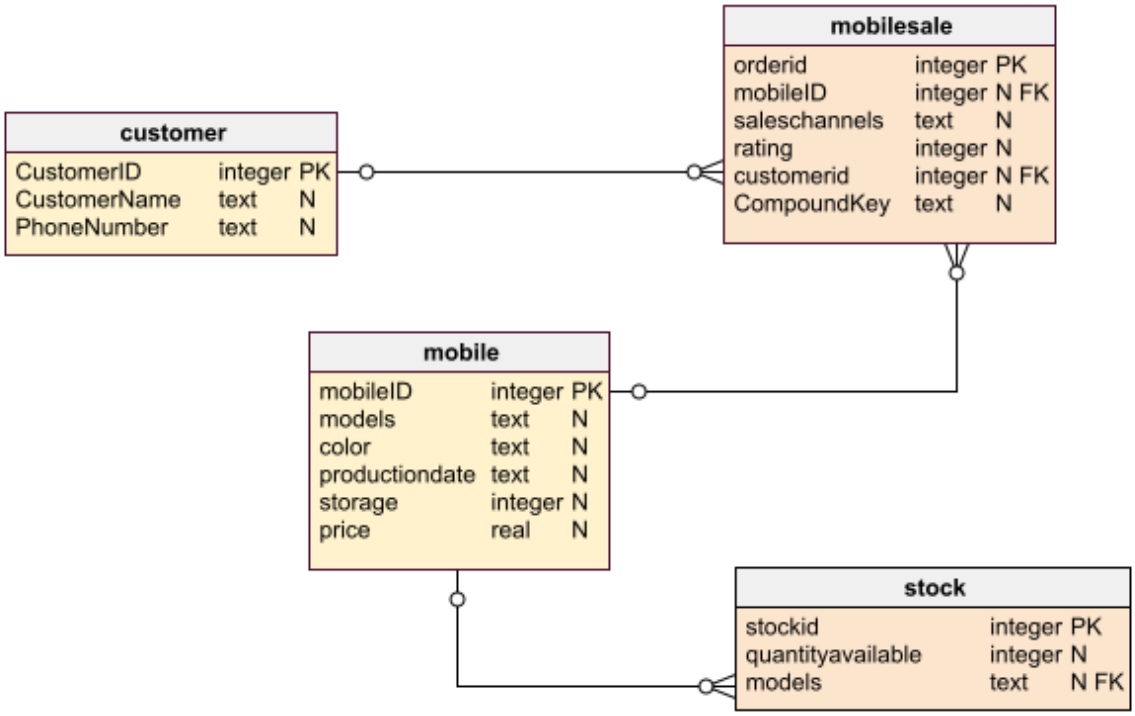
This code creates a dataset for sales on mobile devices and gives each transaction a distinct order ID. The sales channels are assigned at random and indicate if the sale was made "Offline" or "online". In order to

imitate missing data, some ratings are purposefully set to NaN, in an ordinal scale ranging from 2 to 10. The data that is produced is arranged into a dataframe called “df_Mobilescale”, which also has Customer IDs and Mobile IDs. A compound key that combines Order IDs and Mobile IDs is created to demonstrate the relationship between order and mobile. The dataframe is saved as a CSV file called “mobile.csv” and the output is given below.

| order id | mobileID | sales channels | rating | customer id | CompoundKey |
|---|---|---|---|---|---|
| **103** | 8649 | Online | 2.0 | 30794 | 103_8649 |
| **112** | 4923 | Offline | 3.0 | 76513 | 112_4923 |
| **114** | 800 | Offline | 2.0 | 93068 | 114_800 |
| **117** | 6512 | Offline | 4.0 | 47760 | 117_6512 |
| **119** | 2917 | Offline | NaN | 8592 | 119_2917 |
| **...** | ... | ... | ... | ... | ... |
| **9930** | 217 | Offline | 4.0 | 97748 | 9930_217 |
| **9962** | 5814 | Online | 4.0 | 29285 | 9962_5814 |
| **9974** | 6372 | Online | NaN | 58664 | 9974_6372 |
| **9976** | 9296 | Offline | 3.0 | 40612 | 9976_9296 |
| **9985** | 9384 | Offline | 5.0 | 21696 | 9985_9384 |

1000 rows × 5 columns

The schema establishes the relationship between tables and their structure, ensuring a methodical approach to information management. The customer, mobile, mobilesale, and stock tables in the database schema provide a structured framework for organizing and evaluating mobile sales data. The relation connecting the four tables is displayed in the database schema provided below. Examine the relationship between the customer and mobile sale tables. In mobile sale, the customer id serves as a foreign key that displays the specifics of the mobile sale. Furthermore, models serve as a foreign key in the stock table, allowing us to access the models’ stock details.



```sql
SELECT models, rating
FROM mobile
INNER JOIN mobilesale
ON mobile.mobileID = mobilesale.mobileID
WHERE rating > 7 AND productiondate BETWEEN 2020 AND 2022
ORDER BY rating DESC
LIMIT 4;
```

| | models | rating |
|---|---|---|
| 1 | Galaxy Note 20 | 10 |
| 2 | Galaxy XCover Pro | 10 |
| 3 | Galaxy A52 | 10 |
| 4 | Galaxy S21 | 8 |

The query above selects the models which have ratings above 7 and years between 2020 and 2022.

```sql
SELECT customer.CustomerName, customer.customerid, customer.PhoneNumber, mobilesale.saleschannels
FROM customer
JOIN mobilesale
ON customer.customerid = mobilesale.customerid
WHERE saleschannels = 'Online'
ORDER BY customer.customerid
LIMIT 3
```

| | CustomerName | CustomerID | PhoneNumber | saleschannels |
|---|---|---|---|---|
| 1 | Mrs. Pamela Wright | 27990 | +40 5560828985 | Online |
| 2 | Kyle Brown | 61416 | +1 5912530705 | Online |
| 3 | Meredith Stewart | 85489 | +97 1178740457 | Online |

The query above selects all the customer details who are purchased online.

```sql
SELECT models, SUM(price) AS total_revenue
FROM mobile
GROUP BY models;
```

| | models | total_revenue |
|---|---|---|
| 1 | Galaxy A32 | 85221.44 |
| 2 | Galaxy A52 | 79849.19 |
| 3 | Galaxy A71 | 70793.53 |
| 4 | Galaxy M32 | 67929.22 |
| 5 | Galaxy Note 20 | 69462.63 |

The query above calculates the total sum of prices of each model and represents as total revenue.

```sql
SELECT *
FROM mobilesale
WHERE mobilesale.CompoundKey = "136_1782"
```

| | orderid | mobileID | saleschannels | rating | customerid | CompoundKey |
|---|---|---|---|---|---|---|
| 1 | 136 | 1782 | Online | 4 | 22497 | 136_1782 |

The query above is designed to retrieve all the information related to the sale identified by the compound key”136_1782” from the mobilesale table.