Let's install required libraries

```
# pip install sqlalchemy
!pip install sqlalchemy

# pip install pymysql
!pip install pymysql

#importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import sqlalchemy
engine =
sqlalchemy.create_engine('mysql+pymysql://root:root@localhost:3306/
gdb041')
```

Let's import tables from MySQL and load them into Pandas dataframes

```
#importing "fact_sales_monthly" table from MySQL into dataframe
"df_fact_sales_monthly"

df_fs_monthly = pd.read_sql_table('fact_sales_monthly',engine)

print("Shape of dataframe: ", df_fs_monthly.shape)

#top 5 rows
df_fs_monthly.head()
```

```
Shape of dataframe:  (1425706, 5)
```

	date	fiscal_year	product_code	customer_code	sold_quantity
0	2017-09-01	2018	A0118150101	70002017	51
1	2017-09-01	2018	A0118150101	70002018	77
2	2017-09-01	2018	A0118150101	70003181	17
3	2017-09-01	2018	A0118150101	70003182	6
4	2017-09-01	2018	A0118150101	70006157	5

Let's see number of unique categories in columns

```
print("Total no of unique fiscal years: ",
len(df_fs_monthly['fiscal_year'].unique()))

print("Total no of unique product_codes: ",
len(df_fs_monthly['product_code'].unique()))

print("Total no of unique customers: ",
len(df_fs_monthly['customer_code'].unique()))
```

```
Total no of unique fiscal years:  5
Total no of unique product_codes:  389
Total no of unique customers:  209

#let's check for any null values in the data

df_fs_monthly.isnull().sum()

date              0
fiscal_year       0
product_code      0
customer_code     0
sold_quantity     0
dtype: int64
```

- Looks like there are no null values in the data

```
#let's check for any bad /abnormal records in the data

df_fs_monthly[df_fs_monthly['sold_quantity'] == 0]

           date  fiscal_year product_code  customer_code
sold_quantity
12   2017-09-01         2018  A0118150101       70012042
0
13   2017-09-01         2018  A0118150101       70012043
0
48   2017-09-01         2018  A0118150101       90012033
0
49   2017-09-01         2018  A0118150101       90012034
0
50   2017-09-01         2018  A0118150101       90012035
0
...         ...          ...          ...            ...             ..
.
5064 2017-09-01         2018  A7118160101       90012035
0
5065 2017-09-01         2018  A7118160101       90012037
0
5066 2017-09-01         2018  A7118160101       90012038
0
5067 2017-09-01         2018  A7118160101       90012039
0
5068 2017-09-01         2018  A7118160101       90012041
0

[783 rows x 5 columns]
```

Key Finding:

- It makes no sense of having sold_quantity zero in fact_sales_monthly table.

- Having these values in data does not add any value for further exploration.
- Let's remove/filter all these zero sold quantity values.

Filtering the data

```
df_fs_monthly = df_fs_monthly[df_fs_monthly['sold_quantity'] != 0]

print("New shape: ", df_fs_monthly.shape)

New shape:  (1424923, 5)
```

Business Question: 1

- Get the total sold quantity for each fiscal year.
- Show it through visualisation.

```
df_fs_monthly.head()
```

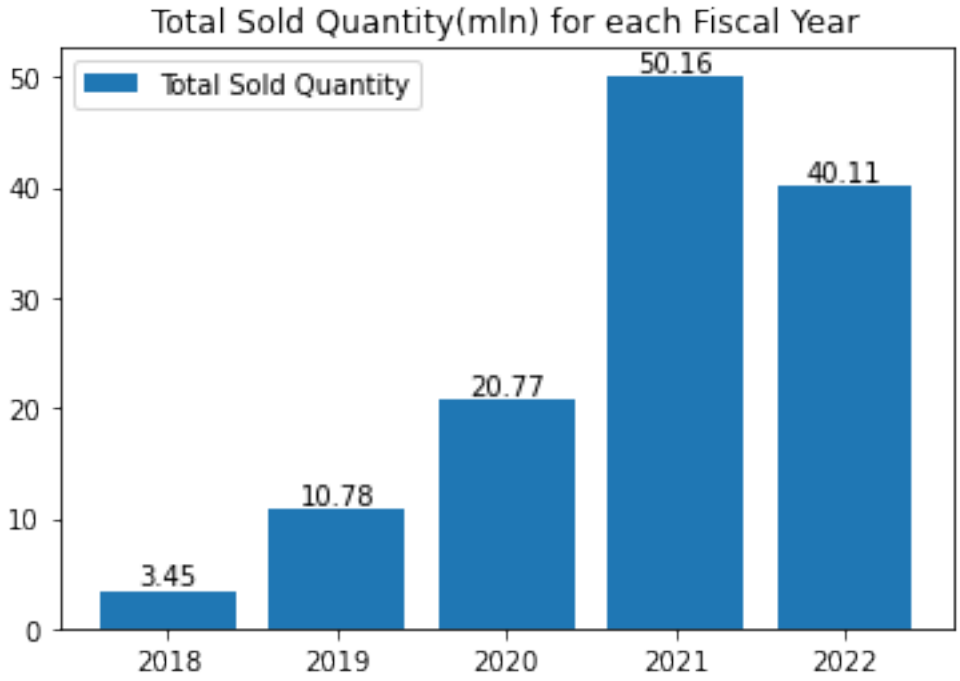
	date	fiscal_year	product_code	customer_code	sold_quantity
0	2017-09-01	2018	A0118150101	70002017	51
1	2017-09-01	2018	A0118150101	70002018	77
2	2017-09-01	2018	A0118150101	70003181	17
3	2017-09-01	2018	A0118150101	70003182	6
4	2017-09-01	2018	A0118150101	70006157	5

```
query = """
    SELECT
        fiscal_year,
        ROUND(SUM(sold_quantity) / 1000000, 2)  as total_qty_sold_mln
    FROM fact_sales_monthly
    GROUP BY fiscal_year;
"""


df_yearly_sales = pd.read_sql_query(query,engine)
df_yearly_sales
```

	fiscal_year	total_qty_sold_mln
0	2018	3.45
1	2019	10.78
2	2020	20.77
3	2021	50.16
4	2022	40.11

```
#showing the data through visualisations

all_years = df_yearly_sales['fiscal_year'].values.tolist()
total_sales_values =
df_yearly_sales['total_qty_sold_mln'].values.tolist()

xpos = np.arange(len(all_years))
plt.bar(xpos, total_sales_values, label="Total Sold Quantity")
```

```
plt.xticks(xpos, all_years)

for i in range(len(total_sales_values)):
    plt.text(i, total_sales_values[i], total_sales_values[i], ha =
"center", va = "bottom")

plt.title(f"Total Sold Quantity(mln) for each Fiscal Year")
plt.legend()
plt.show()
```



Bussiness Insights:

- 1 For every fiscal year, the total sold quantity is growing more than double of its previous year which is very good sign and depicts the business expansion.
- 2 In 2022, total sales had dropped and there is a decline in the bussiness, Note that in 2022, we have data upto december only which is 4th month of 2022 fiscal year and still 8 more months to go and we can expect very high total sales.

Business Question: 2

- In which quarter of 2021, we got the maximum sold quantity.

```
#as Quarter information is not present in our SQL dataset, need to add
derived column using CASE-WHEN statement

query = """
    SELECT
       *,
       GET_quarter(s.date) as quarter
```

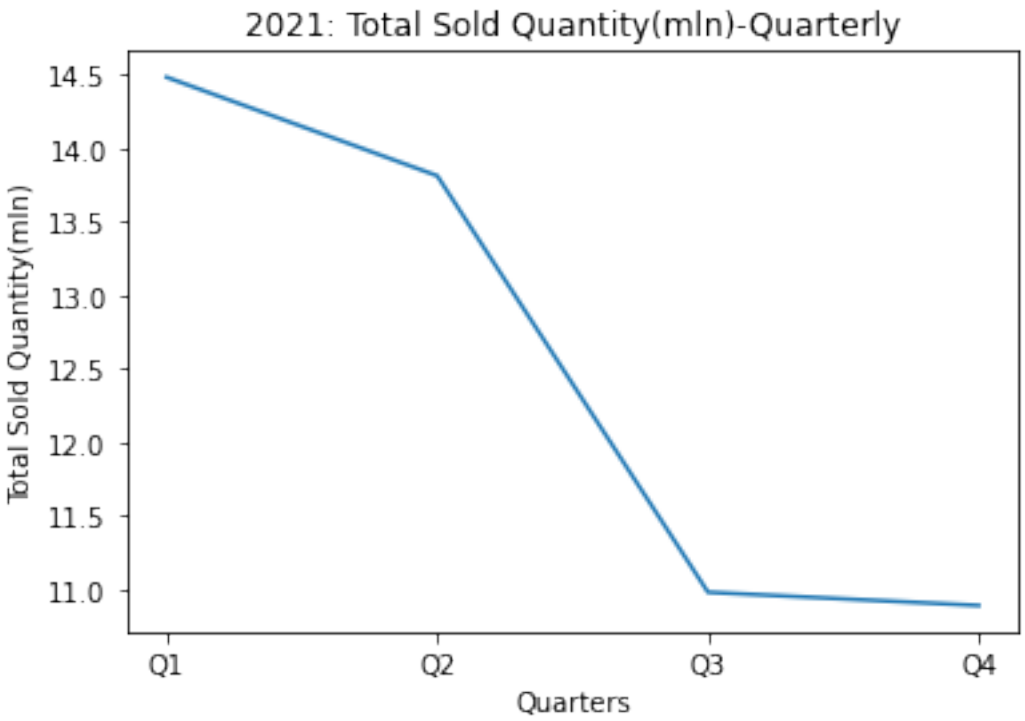
```
        FROM fact_sales_monthly s
    WHERE fiscal_year = 2021
"""

df_quaterly_sales = pd.read_sql_query(query,engine)
df_quaterly_sales.head()
```

	date	fiscal_year	product_code	customer_code	sold_quantity	quarter
0	2020-09-01	2021	A0118150101	70002017	248	Q1
1	2020-09-01	2021	A0118150101	70002018	240	Q1
2	2020-09-01	2021	A0118150101	70003181	31	Q1
3	2020-09-01	2021	A0118150101	70003182	37	Q1
4	2020-09-01	2021	A0118150101	70004069	7	Q1

```
#now let's groupby quarter and get the total sold quantity
quarter_info = pd.DataFrame(df_quaterly_sales.groupby('quarter')
['sold_quantity'].agg(sum).reset_index())

# #let's convert sold quantity to millions for better readability
quarter_info['sold_quantity'] =
quarter_info['sold_quantity'].apply(lambda x: round(x / 1000000, 2))

quarter_info.head()
```

	quarter	sold_quantity
0	Q1	14.48
1	Q2	13.81
2	Q3	10.98
3	Q4	10.89

```
# plot a line graph
plt.plot(quarter_info["quarter"], quarter_info["sold_quantity"])

plt.title(f"2021: Total Sold Quantity(mln)-Quarterly")

plt.xlabel("Quarters")
plt.ylabel("Total Sold Quantity(mln)")
plt.show()
```



Bussiness Insights: 2021

- From the above, we can see that **Q1** has the highest total sales followed by Q2.
- Through investigation, it is found that Quarter1 and Quarter2 has major events across the world like Christmas, Dhussera, Diwali etc which are helping to generate more sales and revenue to Atliq company.
- So, like every fiscal year, need to be more attention in these Quarters and have very good back-up of the products in Warehouses.

Business Question: 3

- Generate a report with Top 5 products in each division according to the total sold quantity in the fiscal year 2021.
- Save that generated report data to .csv file and send to sales director head.

```
df_top_products = pd.read_sql_query(
    f"call gdb041.get_top_n_products_per_division_by_qty_sold(2021,
5);",
    engine
)

df_top_products
```

	division	product	total_qty	drnk
0	N & S	AQ Pen Drive DRC	2034569.0	1
1	N & S	AQ Digit SSD	1240149.0	2
2	N & S	AQ Clx1	1238683.0	3
3	N & S	AQ Neuer SSD	1225985.0	4

```
4     N & S                    AQ Clx2  1201025.0     5
5     P & A               AQ Gamers Ms  2477098.0     1
6     P & A               AQ Maxima Ms  2461991.0     2
7     P & A  AQ Master wireless x1 Ms  2448784.0     3
8     P & A     AQ Master wired x1 Ms  2447468.0     4
9     P & A                 AQ Lite Ms  2443425.0     5
10       PC                   AQ Digit   135092.0     1
11       PC                   AQ Gen Y   135031.0     2
12       PC                   AQ Elite   134431.0     3
13       PC                   AQ Gen X   134264.0     4
14       PC                 AQ Velocity   101757.0     5
```

```
#saving the dataframe

df_top_products.to_csv("Top_products_2021.csv", index = False)
```

Business Question: 4

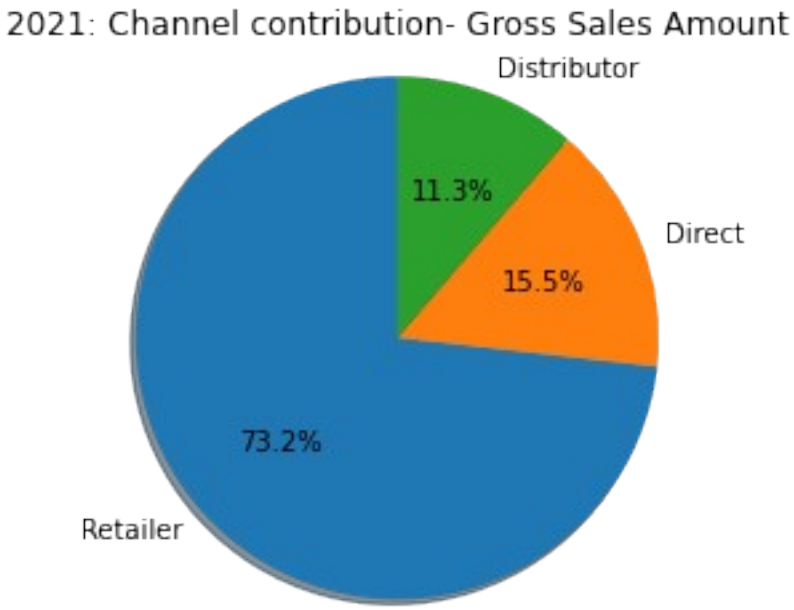
- Which channel helped us to bring more gross sales in the year 2021 and the percentage of contribution.
- Plot a Pie chart to show the visual representation and save it and send it to Sales Manager.

```
query = """
    with channel_gross_sales as
    (
        SELECT
            c.channel as channel,
            ROUND(SUM(s.gross_price_total)/1000000,2) as
gross_sales_mln
        FROM
                gdb041.gross_sales s
        JOIN
                gdb041.dim_customer c USING (customer_code)
        WHERE
                s.fiscal_year = 2021
        GROUP BY channel
    )

    SELECT
        channel,
        gross_sales_mln,
        ROUND(100 * gross_sales_mln / SUM(gross_sales_mln) OVER (),
                2
              ) AS percentage
        From channel_gross_sales
        order by percentage DESC;
"""
```

```
df_channel_gross = pd.read_sql_query(query, engine)
df_channel_gross
```

	channel	gross_sales_mln	percentage
0	Retailer	1219.08	73.23
1	Direct	257.53	15.47
2	Distributor	188.03	11.30

```
#plot the pie chart

data = df_channel_gross['percentage'].values.tolist()
label = df_channel_gross['channel'].values.tolist()

plt.pie(data, labels=label, autopct='%1.1f%%', explode=[0,0,0],
shadow=True, startangle=90)
plt.title('2021: Channel contribution- Gross Sales Amount')
plt.axis('equal')
plt.savefig('channel_contribution (2021).png')  #saving the image
plt.show()
```



Business Insights:

- In 2021, Retailers contribute nearly 73% of total gross_sales amount.
- We can give good pre-invoice deductions(discounts) on products for top performing retailers and that have a scope to maintain good relationships with them and thus have a scope to increase more gross sales.
- We need to think why **Direct(Atliq stores)** are failing to perform same as retailers and do through study of sucess measures of retailers and try to implement for our stores