Atliq Hardware Business Analysis

Table of contents

```

```

1. Data Exploration using Python Pandas

Connect to database and read dim tables

```python
# import necessary libraries
from sqlalchemy import create_engine
import pandas as pd

# Connect with MySql database
engine = create_engine("mysql+pymysql://root:root@localhost:3306/gdb0041")
conn = engine.connect()

# Read in dimension tables
dim_customer = pd.read_sql("dim_customer", conn)
dim_product = pd.read_sql("dim_product", conn)

# Sample rows from dim_customer
dim_customer.sample(3)
```

	customer_code	customer	platform	channel market	sub_zone	region
109	90012040	Fnac-Darty	Brick & Mortar	Retailer	Germany	NE
197	90023024	Sage	Brick & Mortar	Retailer	Canada	NA
121	90014136	Reliance Digital	Brick & Mortar	Retailer	Netherlands	NE

No of unique customers and countries(markets)

```python
# No of unique customers and countries
for column in ["customer", "market"]:
    print(f"No of unique {column}s: {dim_customer[column].nunique()}")
```

```
No of unique customers: 75
```

```
No of unique markets: 27
```

What are different platforms, channels, sub_zones, regions

```python
# different platforms, channels, sub_zones, regions
for column in ["platform", "channel", "sub_zone", "region"]:
    print(f"{column}s -> {dim_customer[column].unique()}")
```

> platforms -> ['Brick & Mortar' 'E-Commerce']

> channels -> ['Direct' 'Distributor' 'Retailer']

> sub_zones -> ['India' 'ROA' 'ANZ' 'SE' 'NE' 'NA' 'LATAM']

> regions -> ['APAC' 'EU' 'NA' 'LATAM']

```
# sample rows from dim_product table
dim_product = pd.read_sql("dim_product", conn)
dim_product.sample(3)
```

	product_code	division	segment	category	product	variant
218	A4319110304	PC	Notebook	Personal Laptop	AQ Velocity	Plus Grey
380	A6818160202	N & S	Storage	USB Flash Drives	AQ Pen Drive DRC	Plus
292	A5318110104	PC	Notebook	Gaming Laptop	AQ Gamer 1	Plus Firey Red

What are different divisions, segments and catogories

```
# Divisions, segments and category
for column in ["division", "segment", "category"]:
    print(f"{column} ({dim_product[column].nunique()}) ->
{dim_product[column].unique()}")
```

> division (3) -> ['P & A' 'PC' 'N & S']

> segment (6) -> ['Peripherals' 'Accessories' 'Notebook' 'Desktop' 'Storage' 'Networking']

> category (14) -> ['Internal HDD' 'Graphic Card' 'Processors' 'MotherBoard' 'Mouse' 'Keyboard' 'Batteries' 'Personal Laptop' 'Business Laptop' 'Gaming Laptop' 'Personal Desktop' 'External Solid State Drives' 'USB Flash Drives' 'Wi fi extender']

```
# no of products per category
dim_product[["category",
"product"]].groupby("category").count().sort_values("product", ascending=False)
```

category	product
Personal Laptop	61
Keyboard	48

category	product
Mouse	48
Business Laptop	44
Gaming Laptop	40
Graphic Card	36
Batteries	20
MotherBoard	20
Processors	18
Personal Desktop	16
External Solid State Drives	15
USB Flash Drives	12
Internal HDD	10
Wi fi extender	9

Read in all the fact tables and give column names and no of rows for each table

```python
# Read in fact tables in database
fact_sales_monthly = pd.read_sql("fact_sales_monthly", conn)
fact_forecast_monthly = pd.read_sql("fact_forecast_monthly", conn)
fact_freight_cost = pd.read_sql("fact_freight_cost", conn)
fact_gross_price = pd.read_sql("fact_gross_price", conn)
fact_manufacturing_cost = pd.read_sql("fact_manufacturing_cost", conn)
fact_pre_invoice_deductions = pd.read_sql("fact_pre_invoice_deductions", conn)
fact_post_invoice_deductions = pd.read_sql("fact_post_invoice_deductions", conn)

# Dictionary of tables and tables names
dict_of_tables = {"fact_sales_monthly": fact_sales_monthly,
"fact_forecast_monthly": fact_forecast_monthly,
                 "fact_freight_cost": fact_freight_cost, "fact_gross_price":
fact_gross_price,
                 "fact_manufacturing_cost": fact_manufacturing_cost,
"fact_pre_invoice_deductions": fact_pre_invoice_deductions,
                 "fact_post_invoice_deductions": fact_post_invoice_deductions}

# A function to print column names and no of rows
def give_columns_nrows(df, name):
    print(name)
    print(f"columns -> {list(df.columns)}")
    print(len(df))
    print("---------------------------------------------------\n")

for key, value in dict_of_tables.items():
    give_columns_nrows(value, key)
```

```
fact_sales_monthly
columns -> ['date', 'product_code', 'customer_code', 'sold_quantity']
1425706
---------------------------------------------------

fact_forecast_monthly
columns -> ['date', 'fiscal_year', 'product_code', 'customer_code',
'forecast_quantity']
1885941
---------------------------------------------------

fact_freight_cost
columns -> ['market', 'fiscal_year', 'freight_pct', 'other_cost_pct']
135
---------------------------------------------------

fact_gross_price
columns -> ['product_code', 'fiscal_year', 'gross_price']
1182
---------------------------------------------------

fact_manufacturing_cost
columns -> ['product_code', 'cost_year', 'manufacturing_cost']
1182
---------------------------------------------------

fact_pre_invoice_deductions
columns -> ['customer_code', 'fiscal_year', 'pre_invoice_discount_pct']
1045
---------------------------------------------------

fact_post_invoice_deductions
columns -> ['customer_code', 'product_code', 'date', 'discounts_pct',
'other_deductions_pct']
2063076
---------------------------------------------------
```

2. Finance Analytics

Task 1: Generate a report of individual product sales for Croma India

Generate a report of individual product sales (aggregated on a monthly basis at the product code level) for Croma India customer for FY=2021. Atliq's fiscal year starts in September. The report should have the following fields.

1. Month
2. Product Name
3. Variant
4. Sold Quantity

5. Gross Price per Item
6. Gross Price total

```sql
-- User defined function to get fiscal year
CREATE FUNCTION `get_fiscal_year` (calendar_date DATE)
    RETURNS INTEGER
DETERMINISTIC
BEGIN
    DECLARE fiscal_year INT;
    SET fiscal_year = YEAR(DATE_ADD(calendar_date, INTERVAL 4 MONTH));
    RETURN fiscal_year;
END

-- filter fact_monthly_sales by customer_id of croma india
WITH cte AS(
    SELECT customer_code
    FROM dim_customer
    WHERE customer LIKE '%croma%' AND market LIKE '%india%'
)
SELECT
    MONTH(s.date) AS month, p.product, p.variant, s.sold_quantity,
    ROUND(g.gross_price, 2) AS gross_price,
    ROUND(s.sold_quantity*g.gross_price, 2) AS gross_price_total
FROM fact_sales_monthly s JOIN dim_product p
ON s.product_code = p.product_code
JOIN fact_gross_price g
ON  g.product_code = s.product_code AND
    g.fiscal_year = get_fiscal_year(s.date)
WHERE
    customer_code = (SELECT * FROM cte) AND
    get_fiscal_year(date) = 2021
ORDER BY date ASC;
```

month	product	variant	sold_quantity	gross_price	gross_price_total
9	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache	Standard	202	19.06	3849.57
9	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache	Plus	162	21.46	3475.95
9	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache	Premium	193	21.78	4203.44
...	...	...	...	...	...

Table exported to a csv file `croma_2021_all_txn.csv`

Task 2: Aggregated monthly gross sales report for Croma

Create aggregated monthly gross sales report for Croma India customer. The report should have following fields.

1. Month
2. Total gross sales amount to chroma india in that month

```
WITH cte AS(
    SELECT customer_code
    FROM dim_customer
    WHERE customer LIKE '%croma%' AND market LIKE '%india%'
)
SELECT
    DATE_FORMAT(s.date, '%m-%Y') AS month,
    SUM(s.sold_quantity*g.gross_price) AS gross_price_total
FROM fact_sales_monthly s JOIN fact_gross_price g
ON  g.product_code = s.product_code AND
    g.fiscal_year = get_fiscal_year(s.date)
WHERE
    customer_code = (SELECT * FROM cte)
GROUP BY 1;
```

> | month | gross_price_total |
> |---|---|
> | 09-2017 | 122407.5582 |
> | 10-2017 | 162687.5716 |
> | 12-2017 | 245673.8042 |
> | ... | ... |

> Table exported to a csv file `croma_monthly_total_sales.csv`

Task 3: Generate a yearly report for Croma India

Generate a yearly report for Croma India where there are two columns

1. Fiscal Year
2. Total Gross Sales amount In that year from Croma

```
WITH cte AS(
    SELECT customer_code
    FROM dim_customer
    WHERE customer LIKE '%croma%' AND market LIKE '%india%'
)
SELECT
    get_fiscal_year(s.date) AS fiscal_year,
    SUM(s.sold_quantity*g.gross_price) AS gross_price_total
FROM fact_sales_monthly s JOIN fact_gross_price g
```

```sql
ON  g.product_code = s.product_code AND
    g.fiscal_year = get_fiscal_year(s.date)
WHERE
    customer_code = (SELECT * FROM cte)
GROUP BY 1;
```

> | fiscal_year | gross_price_total |
> |---|---|
> | 2018 | 1324097.4432 |
> | 2019 | 3555079.0199 |
> | 2020 | 6502181.9143 |
> | 2021 | 23216512.2215 |
> | 2022 | 44638198.9219 |

> Table exported to a csv file `croma_yearly_total_sales.csv`

Task 4: Stored procedure to get monthly gross sales report

Create a stored procedure to get monthly gross sales report for any customer

```sql
CREATE PROCEDURE `get_monthly_gross_sales_for_customer` (c_code INT)
BEGIN
    SELECT
        DATE_FORMAT(s.date, '%m-%Y') AS month,
        ROUND(SUM(s.sold_quantity*g.gross_price),2) AS gross_price_total
    FROM fact_sales_monthly s JOIN fact_gross_price g
    ON  g.product_code = s.product_code AND
        g.fiscal_year = get_fiscal_year(s.date)
    WHERE
        customer_code = c_code
    GROUP BY 1;
END
```

Task 5: Stored procedure to determine the market badge

Create a stored procedure that can determine the market badge based on the following logic.

If *total_sold_quantity* > *5 million* that market is considered *Gold* else it is *Silver*

Input to the stored proc will be:

- market
- fiscal_year

Output

- market_badge

```
CREATE PROCEDURE `get_market_badge`(
    IN in_market VARCHAR(45),
    IN in_fiscal_year YEAR,
    OUT out_badge VARCHAR(20)
)
BEGIN
    DECLARE qty INT DEFAULT 0;

    # set default market to be india
    IF in_market = "" THEN
        SET in_market = "india";
    END IF;

    # retrieve total qty for a given market + fyear
    SELECT SUM(sold_quantity) INTO qty
    FROM fact_sales_monthly s JOIN dim_customer c
    ON s.customer_code = c.customer_code
    WHERE
        get_fiscal_year(s.date) = in_fiscal_year AND
        c.market = in_market
    GROUP BY c.market;

    # determine market badge
    IF qty > 5000000 THEN
        SET out_badge = "Gold";
    ELSE
        SET out_badge = "Silver";
    END IF;
END
```

3. Top Customers, Products, Markets

Task 1: Create a generated column `fiscal_year`

Add a generated column `fiscal_year` to `fact_sales_monthly` table for query optimization and performance improvement.

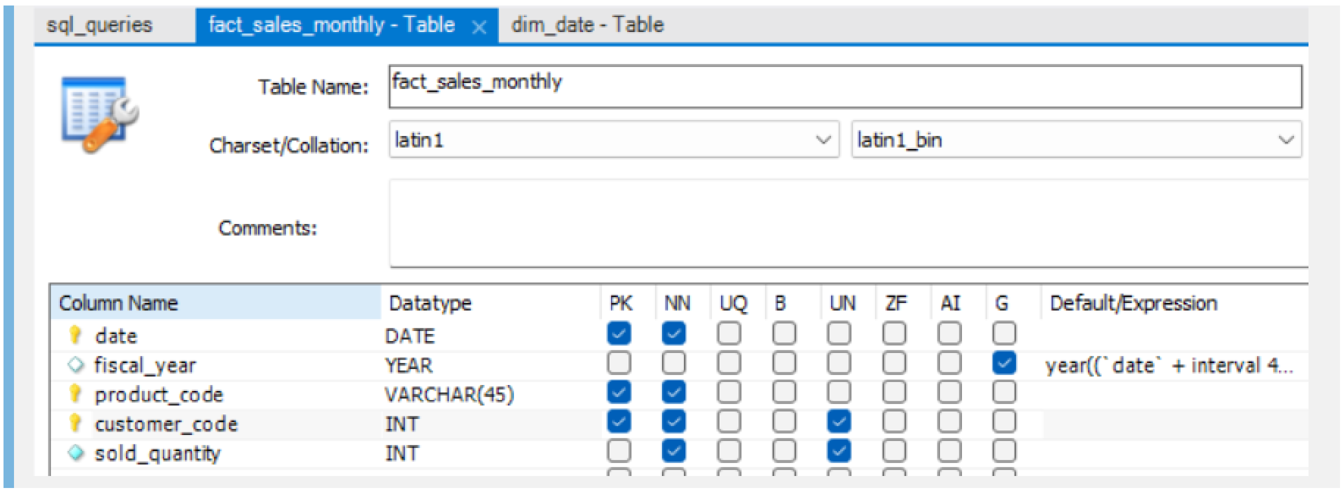


Table Name:	fact_sales_monthly
Charset/Collation:	latin1 / latin1_bin
Comments:	

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
date	DATE	✓	✓							
fiscal_year	YEAR								✓	year((`date` + interval 4...
product_code	VARCHAR(45)	✓	✓							
customer_code	INT	✓	✓			✓				
sold_quantity	INT		✓			✓				

Task 2: Create `database views`

Create following `database views`.

- `sales_preinv_discount`
- `sales_postinv_discount`
- `net_sales`

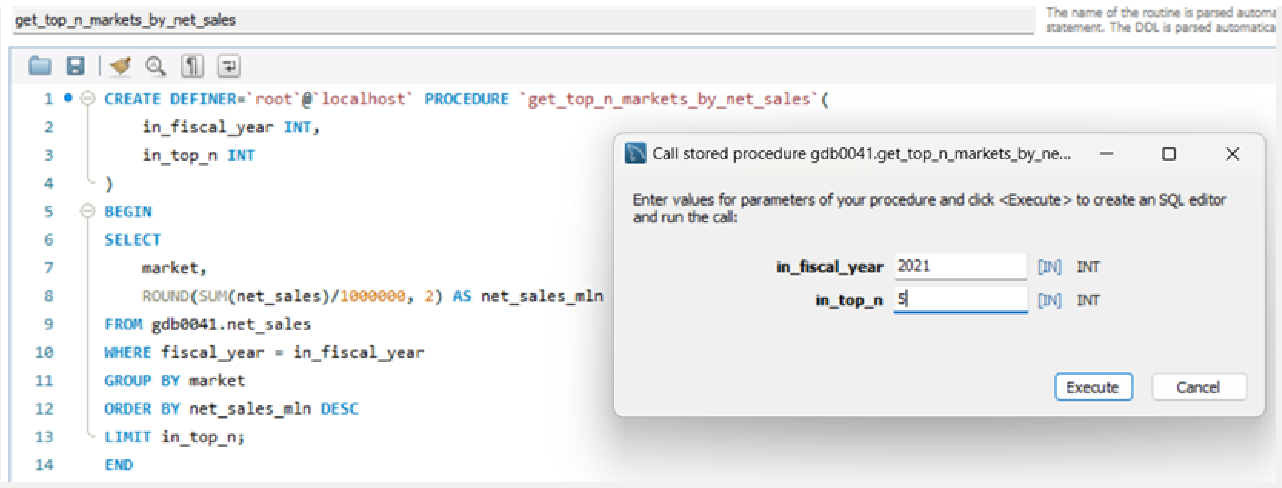
```sql
-- database view #sales_preinv_discount
CREATE VIEW `sales_preinv_discount` AS
SELECT
    s.date, s.fiscal_year,
    s.customer_code, s.market, s.product_code,
    p.product, p.variant,
    s.sold_quantity,
    g.gross_price AS gross_price_per_item,
    ROUND(s.sold_quantity * g.gross_price, 2) AS gross_price_total,
    pre.pre_invoice_discount_pct
FROM fact_sales_monthly s
JOIN dim_customer c
    ON s.customer_code = c.customer_code
JOIN dim_product p
    ON s.product_code = p.product_code
JOIN fact_gross_price g
    ON g.fiscal_year = s.fiscal_year
    AND g.product_code = s.product_code
JOIN fact_pre_invoice_deductions AS pre
    ON pre.customer_code = s.customer_code
    AND pre.fiscal_year = s.fiscal_year


-- database view #sales_postinv_discount
CREATE VIEW sales_postinv_discount AS
SELECT
    s.date, s.fiscal_year,
    s.customer_code, s.market,
    s.product_code, s.product,
    s.variant, s.sold_quantity,
    s.gross_price_total,
    s.pre_invoice_discount_pct,
    (s.gross_price_total - (s.pre_invoice_discount_pct * s.gross_price_total)) AS
net_invoice_sales,
    (po.discounts_pct + po.other_deductions_pct) AS post_invoice_discount_pct
FROM
    sales_preinv_discount s
    JOIN fact_post_invoice_deductions po
        ON po.customer_code = s.customer_code
        AND po.product_code = s.product_code
        AND po.date = s.date

-- database view # net_sales
CREATE VIEW `net_sales` AS
SELECT *,
    (1 - post_invoice_discount_pct)*net_invoice_sales as net_sales
FROM sales_postinv_discount;
```

Task 3: Query top 5 markets by net sales in fy 2021

```
SELECT
    market,
    ROUND(SUM(net_sales)/1000000, 2) AS net_sales_mln
FROM gdb0041.net_sales
WHERE fiscal_year = 2021
GROUP BY market
ORDER BY net_sales_mln DESC
LIMIT 5
```

market	net_sales_mln
India	210.67
USA	132.05
South Korea	64.01
Canada	45.89
United Kingdom	44.73

Also a stored procedure was created which will query *top n markets* in given *fiscal year*



get_top_n_markets_by_net_sales

The name of the routine is parsed automa statement. The DDL is parsed automatica

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_top_n_markets_by_net_sales`(
    in_fiscal_year INT,
    in_top_n INT
)
BEGIN
SELECT
    market,
    ROUND(SUM(net_sales)/1000000, 2) AS net_sales_mln
FROM gdb0041.net_sales
WHERE fiscal_year = in_fiscal_year
GROUP BY market
ORDER BY net_sales_mln DESC
LIMIT in_top_n;
END
```

Call stored procedure gdb0041.get_top_n_markets_by_ne...

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

in_fiscal_year	2021	[IN]	INT
in_top_n	5	[IN]	INT

Execute Cancel

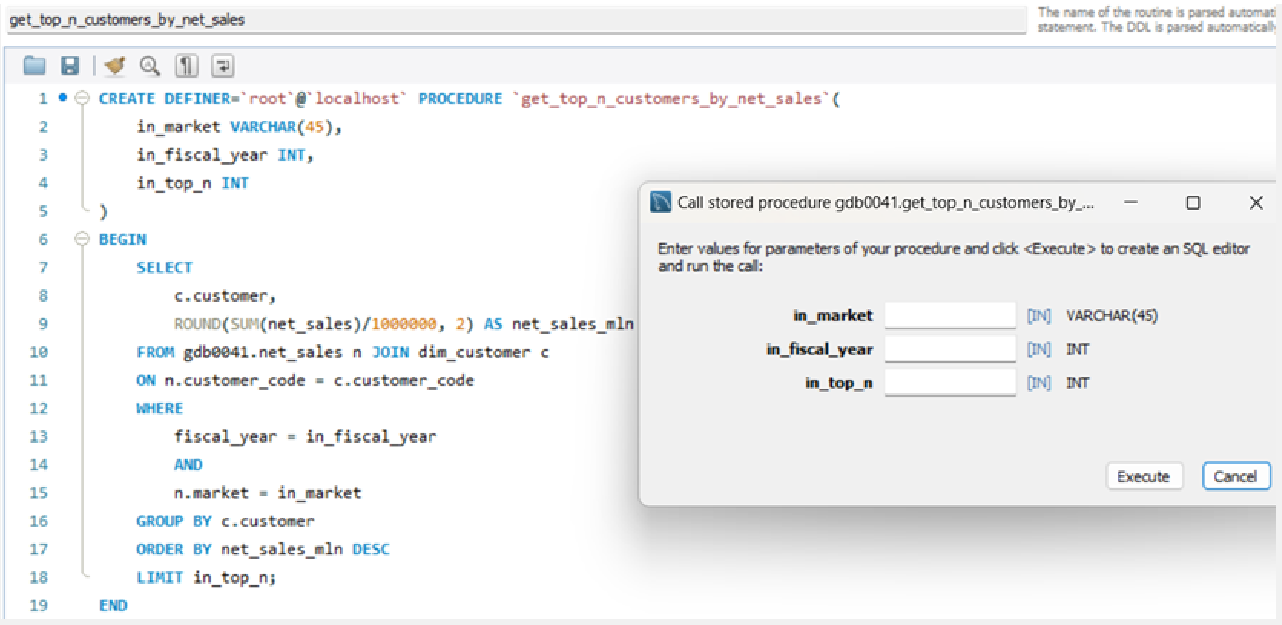
Task 4: Top 5 customers by net_sales for fy 2021

```
SELECT c.customer, ROUND(SUM(net_sales)/1000000, 2) AS net_sales_mln
FROM gdb0041.net_sales n JOIN dim_customer c
ON n.customer_code = c.customer_code
WHERE fiscal_year = 2021
GROUP BY c.customer
```

```
ORDER BY net_sales_mln DESC
LIMIT 5;
```

customer	net_sales_mln
Amazon	109.03
Atliq Exclusive	79.92
Atliq e Store	70.31
Sage	27.07
Flipkart	25.25

Also a stored procedure was created which will query *top n customers* in given *fiscal year* & *market*



get_top_n_customers_by_net_sales

The name of the routine is parsed automat statement. The DDL is parsed automaticall

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_top_n_customers_by_net_sales`(
    in_market VARCHAR(45),
    in_fiscal_year INT,
    in_top_n INT
)
BEGIN
    SELECT
        c.customer,
        ROUND(SUM(net_sales)/1000000, 2) AS net_sales_mln
    FROM gdb0041.net_sales n JOIN dim_customer c
    ON n.customer_code = c.customer_code
    WHERE
        fiscal_year = in_fiscal_year
        AND
        n.market = in_market
    GROUP BY c.customer
    ORDER BY net_sales_mln DESC
    LIMIT in_top_n;
END
```

Call stored procedure gdb0041.get_top_n_customers_by_...

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

in_market		[IN]	VARCHAR(45)
in_fiscal_year		[IN]	INT
in_top_n		[IN]	INT

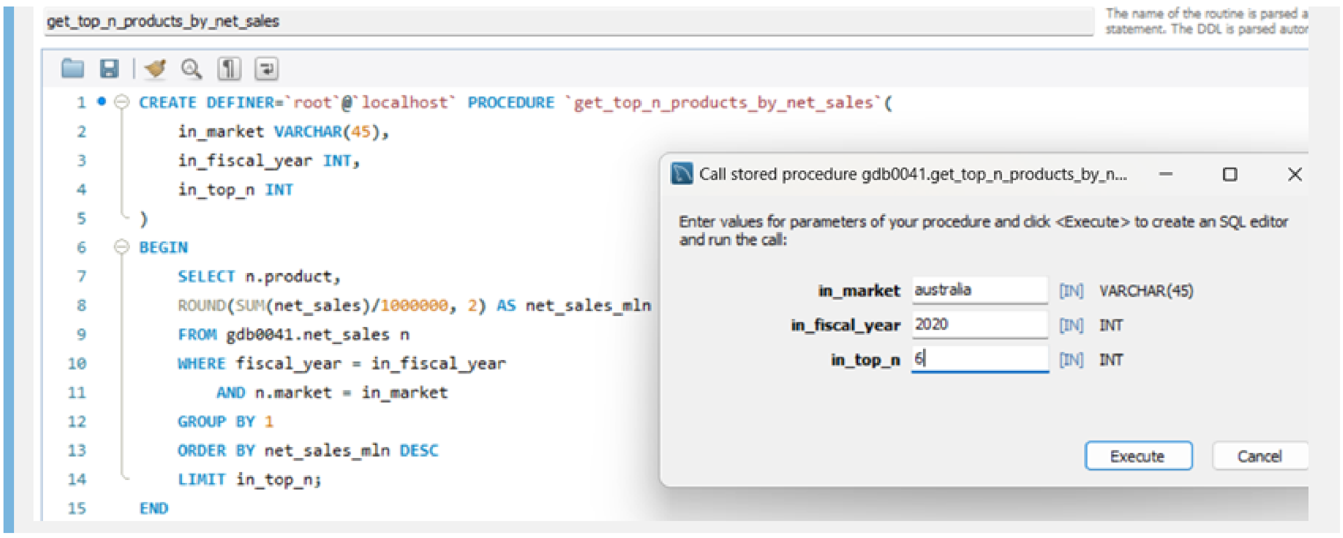
Task 5: Top 5 products by net_sales for fy 2021

```
SELECT n.product, ROUND(SUM(net_sales)/1000000, 2) AS net_sales_mln
FROM gdb0041.net_sales n
WHERE fiscal_year = 2021
GROUP BY 1
ORDER BY net_sales_mln DESC
LIMIT 5;
```

product	net_sales_mln
AQ BZ Allin1	33.75
AQ Qwerty	27.84
AQ Trigger	26.95

product	net_sales_mln
AQ Gen Y	23.58
AQ Maxima	22.32

Also a stored procedure was created which will query *top n products* in given *fiscal year* & *market*



get_top_n_products_by_net_sales

The name of the routine is parsed a
statement. The DDL is parsed autor

```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_top_n_products_by_net_sales`(
    in_market VARCHAR(45),
    in_fiscal_year INT,
    in_top_n INT
)
BEGIN
    SELECT n.product,
    ROUND(SUM(net_sales)/1000000, 2) AS net_sales_mln
    FROM gdb0041.net_sales n
    WHERE fiscal_year = in_fiscal_year
        AND n.market = in_market
    GROUP BY 1
    ORDER BY net_sales_mln DESC
    LIMIT in_top_n;
END
```

Call stored procedure gdb0041.get_top_n_products_by_n...

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

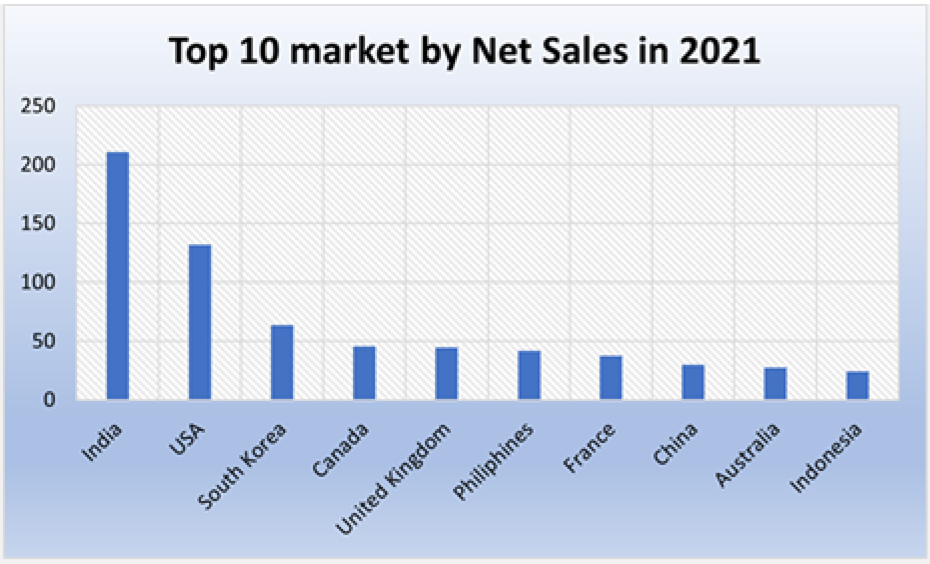
in_market	australia	[IN]	VARCHAR(45)
in_fiscal_year	2020	[IN]	INT
in_top_n	6	[IN]	INT

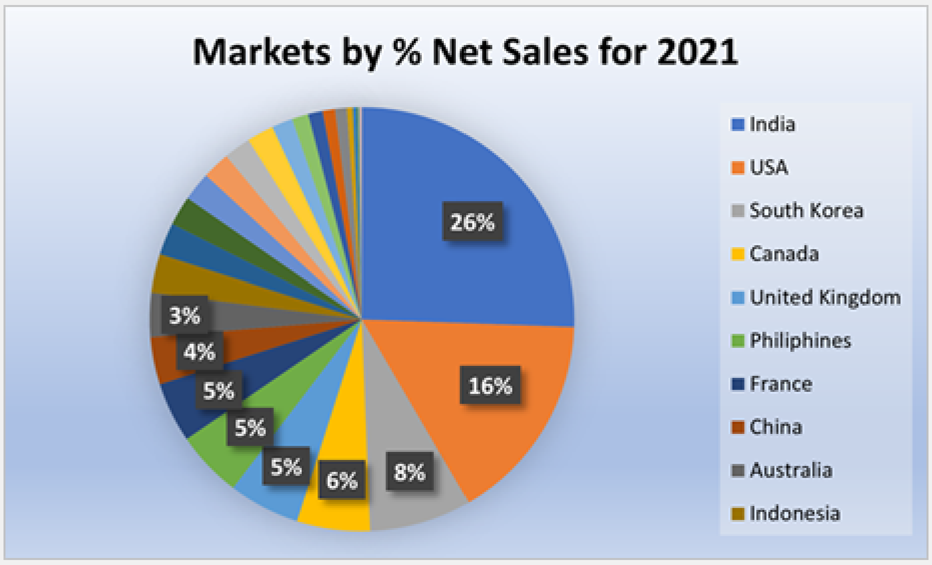
Task 6: Bar Chart report for fy 2021 for top 10 markets & pie chart for % net sales

```sql
SELECT n.market,
        ROUND(SUM(net_sales)/1000000, 2) AS net_sales_mln
FROM gdb0041.net_sales n
WHERE fiscal_year = 2021
GROUP BY 1
ORDER BY net_sales_mln DESC
```

market	net_sales_mln
India	210.67
USA	132.05
South Korea	64.01
Canada	45.89
United Kingdom	44.73
...	...

> This report was exported to excel to create charts





Task 7: Region-wise breakdown of net sales by customers

Create region wise (APAC, EU, LTAM etc) % net sales breakdown by customers in a respective region so regional analysis can be performed.

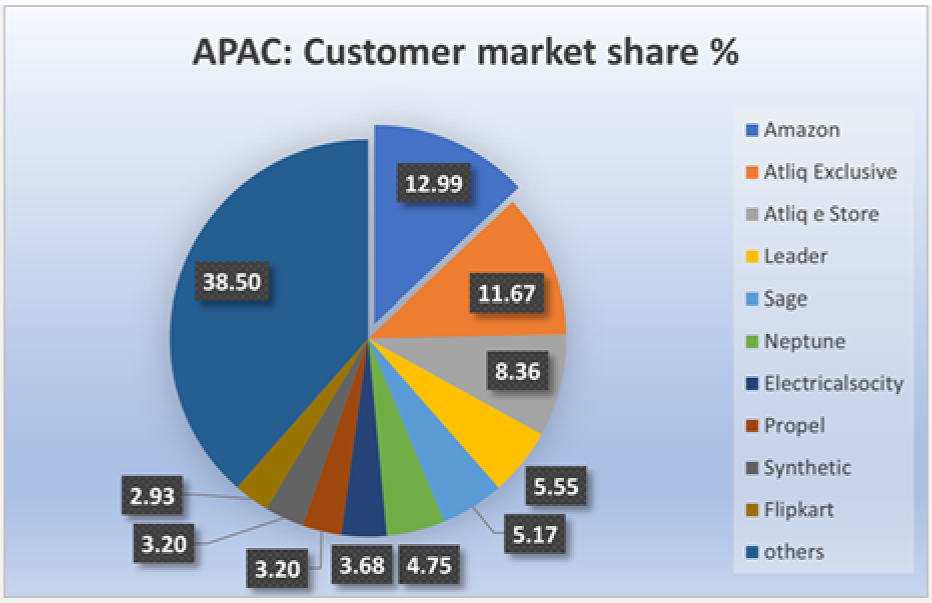
The end result should be Pie Chart in the following format for fy 2021. Build a reusable asset that we can use to conduct this analysis for any financial year.

```
WITH cte AS(
SELECT
    c.customer,
    c.region,
    ROUND(SUM(net_sales)/1000000, 2) AS net_sales_mln
FROM net_sales s JOIN dim_customer c
ON s.customer_code = c.customer_code
WHERE s.fiscal_year = 2021
GROUP BY 1, 2)
SELECT
    *,
    net_sales_mln*100/SUM(net_sales_mln) OVER(PARTITION BY region) AS
pct_share_region
```

```
FROM cte
ORDER BY region, pct_share_region DESC;
```

customer	region	net_sales_mln	pct_share_region
Amazon	APAC	57.41	12.988688
Atliq Exclusive	APAC	51.58	11.669683
Atliq e Store	APAC	36.97	8.364253
Leader	APAC	24.52	5.547511
...	...	...	...

> This report was exported to excel to create charts



Task 8: Retrieve the top 2 markets in every region by their gross sales amount in FY 2021

```
WITH cte1 AS(
    SELECT
        c.market,
        c.region,
        ROUND(SUM(gross_price_total)/1000000, 2) AS gross_sales_mln
    FROM net_sales s JOIN dim_customer c
    ON s.customer_code = c.customer_code
    WHERE s.fiscal_year = 2021
    GROUP BY 1, 2),
cte2 AS(
    SELECT
        *,
        RANK() OVER(PARTITION BY region ORDER BY gross_sales_mln DESC) rn
    FROM cte1)
SELECT *
FROM cte2
WHERE rn <= 2;
```

market	region	gross_sales_mln	rn
India	APAC	455.05	1
South Korea	APAC	131.86	2
United Kingdom	EU	78.11	1
France	EU	67.62	2
Mexico	LATAM	2.30	1
Brazil	LATAM	2.14	2
USA	NA	264.46	1
Canada	NA	89.78	2

4. Ad-hoc Queries, Data Visualization & Business Insights (using Python Pandas)

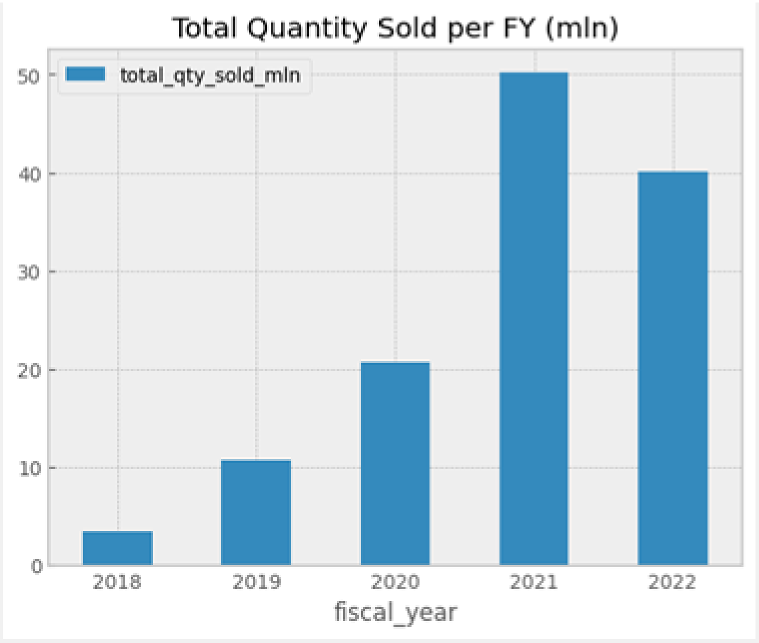
Task 1: Get the total sold quantity for each fiscal year & Show it through visualisation

```
# import necessary libraries
import pandas as pd
from sqlalchemy import create_engine

# connect with database engine
engine = create_engine("mysql+pymysql://root:root@localhost:3306/gdb0041")
conn = engine.connect()

# query total sold quantity per fiscal year
query = """
    SELECT
        fiscal_year,
        ROUND(SUM(sold_quantity) / 1000000, 2)  as total_qty_sold_mln
    FROM fact_sales_monthly
    GROUP BY fiscal_year;
"""

df_sold_qty = pd.read_sql_query(text(query), conn)
df_sold_qty
```

-	fiscal_year	total_qty_sold_mln
0	2018	3.45
1	2019	10.78
2	2020	20.77

-	fiscal_year	total_qty_sold_mln
3	2021	50.16
4	2022	40.11

```
# bar plot for total quantity sold
plt.style.use('bmh')
df_sold_qty.plot(kind="bar", x="fiscal_year", y="total_qty_sold_mln", rot=0,
title="Total Quantity Sold per FY (mln)");
```



🔎 **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, we observe decline in sold qty but, 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.

Task 2: Query the quarters of 2021 by sold quantity and plot line graph

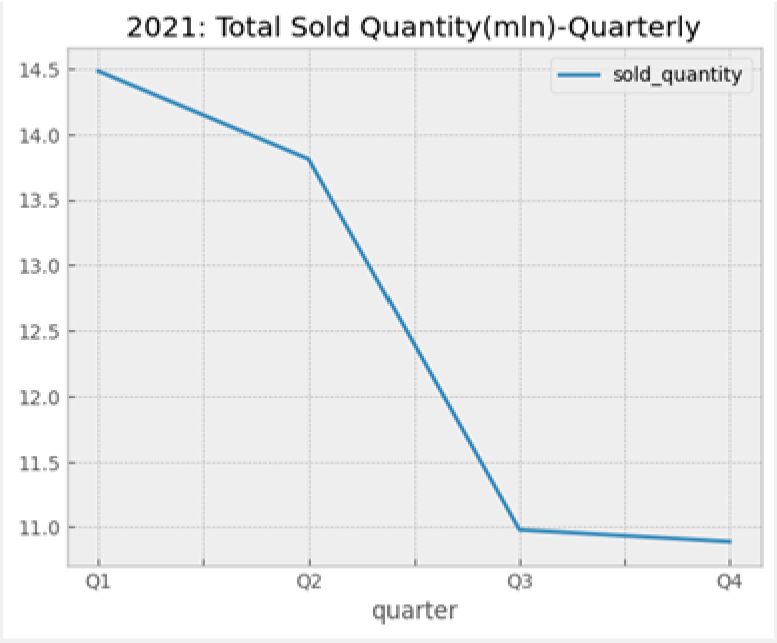
```
# querying the quarters by sold quantity
query = """
    SELECT
       *,
       get_fiscal_quarter(s.date) as quarter
    FROM fact_sales_monthly s
    WHERE fiscal_year = 2021
"""

df_quaterly_sales = pd.read_sql_query(text(query), conn)
df_quaterly_sales.head(2)
```

-	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

```
# Now let's groupby by quarter and convert to million
q = pd.DataFrame(round(df_quaterly_sales.groupby("quarter")
["sold_quantity"].sum()/1000000, 2))
q
```

quarter	sold_quantity
Q1	14.48
Q2	13.81
Q3	10.98
Q4	10.89

```
# Plot sold_quantity on line graph
q.plot(kind="line", y="sold_quantity", title="2021: Total Sold Quantity(mln)-
Quarterly");
```



Bussiness Insights:

- 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.

Task 3: Generate a report with Top 5 products in each division by sold quantity

```
# call a stored procedure
df_top_products = pd.read_sql_query(
    text("call gdb0041.get_top_n_products_per_division_by_qty_sold(2021, 5);"),
conn
)

df_top_products
```

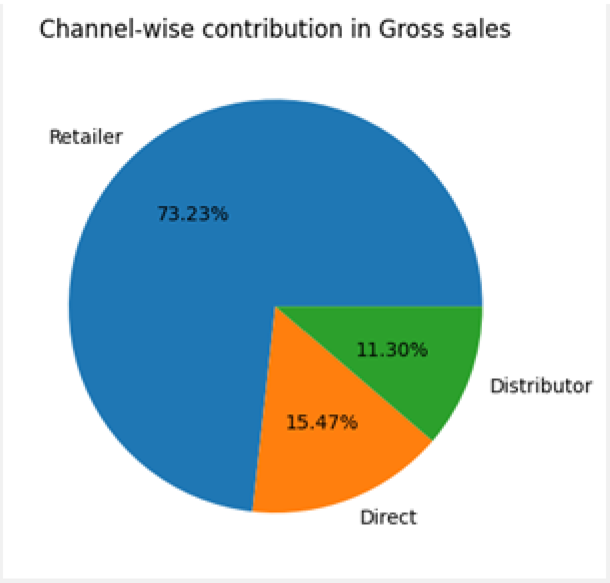
-	division	product	sold_quantity_mln	rnk
0	N & S	AQ Pen Drive DRC	2.0346	1
1	N & S	AQ Digit SSD	1.2401	2
2	N & S	AQ Clx1	1.2387	3
3	N & S	AQ Neuer SSD	1.2260	4
4	N & S	AQ Clx2	1.2010	5
5	P & A	AQ Gamers Ms	2.4771	1
6	P & A	AQ Maxima Ms	2.4620	2
7	P & A	AQ Master wireless x1 Ms	2.4488	3
8	P & A	AQ Master wired x1 Ms	2.4475	4
9	P & A	AQ Lite Ms	2.4434	5
10	PC	AQ Digit	0.1351	1
11	PC	AQ Gen Y	0.1350	2
12	PC	AQ Elite	0.1344	3
13	PC	AQ Gen X	0.1343	4
14	PC	AQ Velocity	0.1018	5

Task 4: Which channel brings more gross sales in the year 2021 and plot the pie chart

```
# query channelwise gross sales
query = """
    WITH channel_gross_sales AS
    (
        SELECT
            c.channel AS channel,
            ROUND(SUM(s.gross_price_total)/1000000,2) AS gross_sales_mln
        FROM
                gdb0041.net_sales s
        JOIN
                gdb0041.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(text(query), conn)
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

```
# Create a Pie chart for channel-wise gross sales
plt.style.use("default")
x = pd.Series(list(df_channel_gross.gross_sales_mln),
index=df_channel_gross.channel)
x.plot(kind="pie", autopct="%.2f%%", title="Channel-wise contribution in Gross
sales");
```



🔎 **Bussiness 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