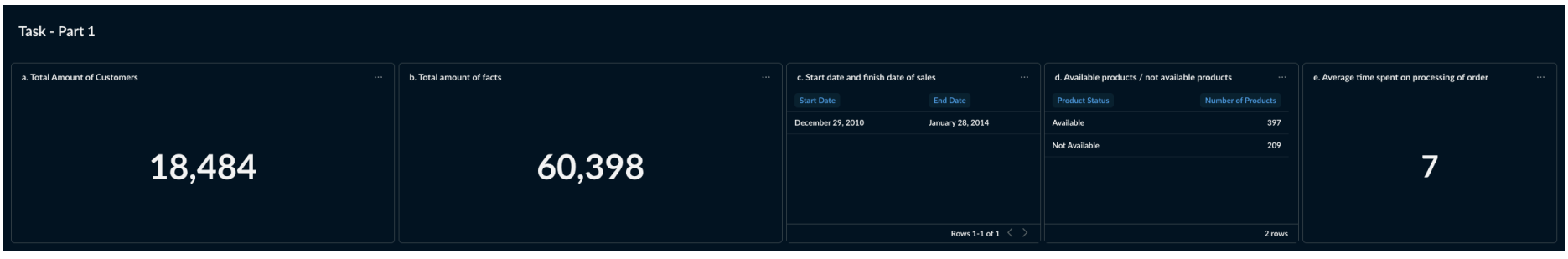
BI-E06-Visualization (Dashboard)

Coban, Omer Furkan

Part 1



a. Total amount of customers

```
SELECT
   COUNT(DISTINCT customer_key) AS total_customers
FROM dim_customer;
```

b. Total amount of facts

```
SELECT
   COUNT(*) AS total_facts
FROM fact_internet_sales;
```

c. Start date and finish date of sales

```
SELECT
   MIN(d.full_date_alternate_key::date) AS start_date,
   MAX(d.full_date_alternate_key::date) AS end_date
FROM fact_internet_sales f
JOIN dim_date d
   ON f.order_date_key = d.date_key;
```

d. Available products / not available products

```
SELECT
  CASE
    WHEN finished_goods_flag = 1 THEN 'Available'
    ELSE 'Not Available'
  END AS product_status,
  COUNT(*) AS number_of_products
FROM
  dim_product
GROUP BY
  product_status;
```

e. Average time spent on processing of order

```
SELECT
   AVG(
      d_ship.full_date_alternate_key::date
    - d_order.full_date_alternate_key::date
   ) AS avg_processing_days
FROM fact_internet_sales f
JOIN dim_date d_order
   ON f.order_date_key = d_order.date_key
JOIN dim_date d_ship
   ON f.ship_date_key = d_ship.date_key
WHERE f.ship_date_key IS NOT NULL;
```

Part 2

1. Visualize on different plots revenue aggregated by the following time dimensions

1. a. Week

```
SELECT
    d.calendar_year AS year,
    d.week_number_of_year AS week,
    SUM(f.sales_amount) AS revenue
FROM fact_internet_sales f
JOIN dim_date d
    ON f.order_date_key = d.date_key
GROUP BY d.calendar_year, d.week_number_of_year
ORDER BY d.calendar_year, d.week_number_of_year;
```

1. b. Month

```
SELECT
    d.calendar_year AS year,
    d.month_number_of_year AS month,
    SUM(f.sales_amount) AS revenue
FROM fact_internet_sales f
JOIN dim_date d
    ON f.order_date_key = d.date_key
GROUP BY d.calendar_year, d.month_number_of_year
ORDER BY d.calendar_year, d.month_number_of_year;
```

1. c. Year

```
SELECT
    d.calendar_year AS year,
    SUM(f.sales_amount) AS revenue
FROM fact_internet_sales f
JOIN dim_date d
    ON f.order_date_key = d.date_key
GROUP BY d.calendar_year
ORDER BY d.calendar_year;
```

2. Create plot that compares two selected weeks bases on weekdays (e.g. calendar week 14 vs calendar week 15, etc.)

```
SELECT
    d.calendar_year AS sales_year,
    d.week_number_of_year AS week,
    d.day_number_of_week,
```

```
    d.english_day_name_of_week AS weekday,
    SUM(f.sales_amount) AS revenue
FROM fact_internet_sales f
JOIN dim_date d
    ON f.order_date_key = d.date_key
WHERE d.calendar_year = 2013
  AND d.week_number_of_year IN (14, 15)
GROUP BY
    sales_year,
    d.week_number_of_year,
    d.day_number_of_week,
    d.english_day_name_of_week
ORDER BY
    d.day_number_of_week,
    d.week_number_of_year;
```

3. Create plot that compares two months from the different years (e.g. May 2012 vs May 2013)

```
SELECT
    d.calendar_year AS year,
    d.day_number_of_month AS day_of_month,
    SUM(f.sales_amount) AS revenue
FROM fact_internet_sales f
JOIN dim_date d
    ON f.order_date_key = d.date_key
WHERE d.month_number_of_year = 5
  AND d.calendar_year IN (2012, 2013)
GROUP BY
    d.calendar_year,
    d.day_number_of_month
ORDER BY
    d.day_number_of_month,
    d.calendar_year;
```

4. Create plot that shows revenue for top 10 products from of a pre-selected year (e.g. 2013)

```
SELECT
    p.english_product_name AS product,
    SUM(f.sales_amount) AS revenue
FROM fact_internet_sales f
JOIN dim_date d
    ON f.order_date_key = d.date_key
JOIN dim_product p
    ON f.product_key = p.product_key
WHERE d.calendar_year = 2013
GROUP BY p.english_product_name
ORDER BY revenue DESC
LIMIT 10;
```

5. Display top 10 products with processing time, which is bigger or equal than average time spent on processing for all products (see order_date and ship_date)

```
WITH order_processing AS (
    SELECT
        f.product_key,
        d_ship.full_date_alternate_key::date
      - d_order.full_date_alternate_key::date AS processing_days
    FROM fact_internet_sales f
    JOIN dim_date d_order
        ON f.order_date_key = d_order.date_key
    JOIN dim_date d_ship
        ON f.ship_date_key = d_ship.date_key
    WHERE f.ship_date_key IS NOT NULL
),
product_avg AS (
    SELECT
        product_key,
        AVG(processing_days) AS avg_processing_days
    FROM order_processing
    GROUP BY product_key
```

```
),
overall_avg AS (
    SELECT
        AVG(processing_days) AS overall_avg_processing_days
    FROM order_processing
)
SELECT
    p.english_product_name AS product,
    pa.avg_processing_days
FROM product_avg pa
JOIN overall_avg oa
    ON pa.avg_processing_days >= oa.overall_avg_processing_days
JOIN dim_product p
    ON pa.product_key = p.product_key
ORDER BY pa.avg_processing_days DESC
LIMIT 10;
```

Part 3

1 . a. Define customers profiles based on following fields - income, married/non-married, children, gender, education, house owners, car owners

```
SELECT
    gender,
    marital_status,
    english_education AS education,
    house_owner_flag,
    CASE
        WHEN yearly_income < 40000 THEN 'Low income'
        WHEN yearly_income BETWEEN 40000 AND 80000 THEN 'Middle income'
        ELSE 'High income'
    END AS income_group,
    CASE
        WHEN number_children_at_home = 0 THEN 'No children'
        WHEN number_children_at_home = 1 THEN '1 child'
        ELSE '2+ children'
    END AS children_group,
    CASE
        WHEN number_cars_owned = 0 THEN 'No car'
        WHEN number_cars_owned = 1 THEN '1 car'
        ELSE '2+ cars'
```

```
    END AS car_group,
    COUNT(*) AS customers,
    AVG(yearly_income) AS avg_income
FROM dim_customer
GROUP BY
    gender,
    marital_status,
    english_education,
    house_owner_flag,
    income_group,
    children_group,
    car_group
ORDER BY customers DESC;
```

1. b. Present each of your customer profiles (at least you should define 3) on your dashboard

```
SELECT
    COUNT(*) AS customers,
    AVG(yearly_income) AS avg_income
FROM dim_customer
WHERE marital_status = 'M'
  AND yearly_income BETWEEN 40000 AND 80000
  AND number_children_at_home >= 1
  AND house_owner_flag = 1;

  SELECT
    COUNT(*) AS customers,
    AVG(yearly_income) AS avg_income
FROM dim_customer
WHERE marital_status = 'S'
  AND yearly_income > 80000
  AND number_children_at_home = 0
  AND english_education IN ('Graduate Degree', 'Bachelors')
  AND number_cars_owned >= 2;

  SELECT
    COUNT(*) AS customers,
    AVG(yearly_income) AS avg_income
FROM dim_customer
WHERE marital_status = 'S'
  AND yearly_income < 40000
  AND number_children_at_home = 0
  AND house_owner_flag = 0;
```

1 . c. Think about, how customer profiles could be potentially used for designing and answering business questions

Customer profiles enable businesses to move from descriptive analytics to actionable insights. By segmenting customers based on demographic and socio-economic attributes, organizations can design targeted business questions related to marketing, pricing, operations, and strategy, ultimately supporting better decision-making.

1. d. Define 2 business questions, which should involve usage of the customer profiles

a) **Which customer profiles generate the highest revenue per customer?**

b) **How does revenue contribution differ across customer profiles?**

1. e. Answer your 2 business questions using visualizations and put them on your dashboard

a. High-income customer profiles generate the highest revenue per customer. This indicates that although this group may be smaller in size, individual customers contribute significantly more revenue compared to middle- and low-income segments.

b. Middle-income customer profiles contribute the largest share of total revenue. While high-income customers generate higher revenue per customer, middle-income customers dominate overall revenue due to their larger population size.

2. Could you use your customer profiles to identify a good / middle / bad customers?

2. a. Define your own KPI to identify good / middle / bad customers. You should describe your own KPI as a text.

Customer Value Score (CVS): The Customer Value Score (CVS) is a composite KPI designed to classify customers into good, middle, and bad segments based on their economic contribution. The KPI combines revenue per customer and purchase frequency to reflect both customer value and engagement. Customers with high spending and frequent purchases are classified as good customers, while customers with low contribution and low engagement are classified as bad customers.

2. b. Provide an exact calculation approach for own KPI in order to demonstrate, how it should work (e.g., SQL query, meth equations, etc.)

$$\text{CVS}_i = 0.6 \times \text{Normalized Revenue}_i + 0.4 \times \text{Normalized Orders}_i$$

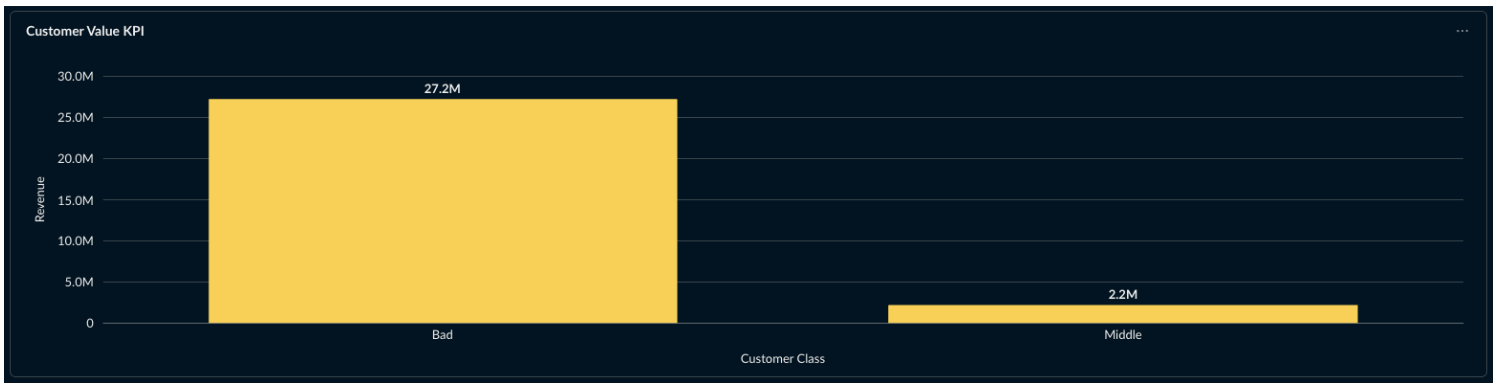
$$\text{Normalized Metric} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

```
WITH customer_metrics AS (
    SELECT
        c.customer_key,
        SUM(f.sales_amount) AS total_revenue,
        COUNT(f.sales_order_number) AS total_orders
    FROM dim_customer c
    LEFT JOIN fact_internet_sales f
        ON c.customer_key = f.customer_key
    GROUP BY c.customer_key
),
normalized AS (
    SELECT
        customer_key,
        total_revenue,
        total_orders,
        (total_revenue - MIN(total_revenue) OVER ())
        / NULLIF(MAX(total_revenue) OVER () - MIN(total_revenue) OVER (), 0)
        AS norm_revenue,
        (total_orders - MIN(total_orders) OVER ())
        / NULLIF(MAX(total_orders) OVER () - MIN(total_orders) OVER (), 0)
        AS norm_orders
    FROM customer_metrics
),
kpi AS (
    SELECT
        customer_key,
        0.6 * norm_revenue + 0.4 * norm_orders AS customer_value_score
    FROM normalized
)
SELECT
    customer_key,
    customer_value_score,
    CASE
        WHEN customer_value_score >= 0.66 THEN 'Good'
        WHEN customer_value_score >= 0.33 THEN 'Middle'
        ELSE 'Bad'
    END AS customer_class
FROM kpi;
```

2. c. Apply your own KPI to visualize on your dashboard good / middle / bad customers

```
SELECT
    customer_class,
    SUM(total_revenue) AS revenue
FROM (
    -- KPI ana sorgusu
    WITH customer_base AS (
        SELECT
            c.customer_key,
            COALESCE(SUM(f.salesamount), 0) AS total_revenue,
            COUNT(f.salesorder_number) AS total_orders
        FROM dim_customer c
        LEFT JOIN fact_internet_sales f
            ON c.customer_key = f.customer_key
        GROUP BY c.customer_key
    ),
    normalized AS (
        SELECT
            customer_key,
            total_revenue,
            (total_revenue - MIN(total_revenue) OVER ())
            / NULLIF(MAX(total_revenue) OVER () - MIN(total_revenue) OVER (), 0)
```

```
        AS norm_revenue,
        (total_orders - MIN(total_orders) OVER ())
        / NULLIF(MAX(total_orders) OVER () - MIN(total_orders) OVER (), 0)
        AS norm_orders
    FROM customer_base
  )
  SELECT
    customer_key,
    total_revenue,
    CASE
      WHEN (0.6 * norm_revenue + 0.4 * norm_orders) >= 0.66 THEN 'Good'
      WHEN (0.6 * norm_revenue + 0.4 * norm_orders) >= 0.33 THEN 'Middle'
      ELSE 'Bad'
    END AS customer_class
  FROM normalized
) t
GROUP BY customer_class
ORDER BY revenue DESC;
```



The resulting distribution shows a strong dominance of low-value customers, with only a very small number of middle-value customers and no customers classified as high-value. This outcome reflects the highly skewed customer value distribution in the dataset and indicates that only a limited number of customers combine both high spending and frequent purchasing behavior.