****

**Interview Test**

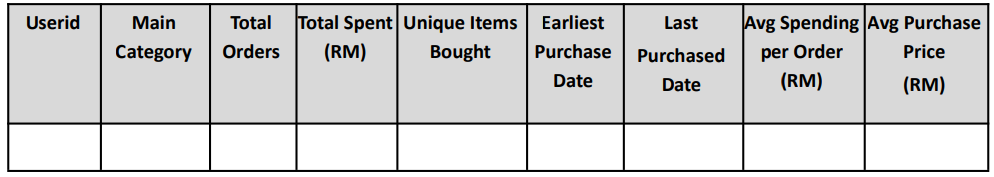
**Senior Business Intelligence**

**Part 1: Forming SQL Queries**

**Q1:** Find the lifetime total orders, total spent (gmv), unique items bought, earliest purchase date, last purchased date, average amount spent per order and average purchase price for the following buyer IDs and their purchased products’ main categories:

* 576123
* 123152

Expected output:



**Solution:**

SELECT

    buyer\_id,

    COUNT(DISTINCT order\_id) AS total\_orders,

    SUM(gmv) AS total\_spent,

    COUNT(DISTINCT product\_id) AS unique\_items\_bought,

    MIN(order\_date) AS earliest\_purchase\_date,

    MAX(order\_date) AS last\_purchased\_date,

    SUM(gmv) / COUNT(DISTINCT order\_id) AS average\_amount\_spent\_per\_order,

    (SUM(gmv) - SUM(shipping\_fee\_paid)) / SUM(qty\_sold) AS average\_purchase\_price,

    l1\_cat AS main\_category

FROM order\_trans

JOIN buyer\_profile ON order\_trans.buyer\_id = buyer\_profile.buyer\_id

WHERE buyer\_id IN (576123, 123152)

GROUP BY buyer\_id, l1\_cat

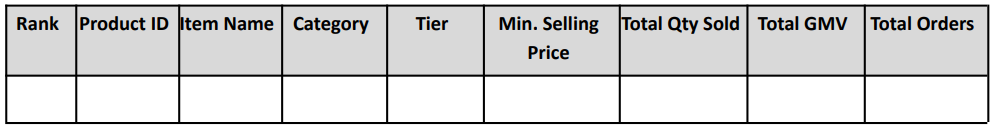
ORDER BY buyer\_id, main\_category;

**Q2:** Find out the top 10 cross border items with the highest quantity sold last month, together with their tier \*\*, minimum selling price, total spent (gmv) and total orders.

\*\* Tier is a user-defined item attribute dimension with 3 unique values:

* Short Tail (>20 average daily orders)
* Mid Tail (between 10 - 20 average daily orders)
* Long Tail (< 10 average daily orders)

**Expected output:**



**Solution:**

WITH item\_tiers AS (

    SELECT

        product\_id,

        CASE

            WHEN AVG(qty\_sold) OVER (PARTITION BY product\_id) > 20 THEN 'Short Tail'

            WHEN AVG(qty\_sold) OVER (PARTITION BY product\_id) BETWEEN 10 AND 20 THEN 'Mid Tail'

            ELSE 'Long Tail'

        END AS tier

    FROM order\_trans

    WHERE is\_cross\_border = 1

)

SELECT TOP 10

    product\_id,

    tier,

    MIN(price) AS minimum\_selling\_price,

    SUM(gmv) AS total\_spent,

    COUNT(DISTINCT order\_id) AS total\_orders

FROM order\_trans

JOIN item\_tiers USING (product\_id)

WHERE is\_cross\_border = 1

AND order\_date >= DATE\_TRUNC('month', CURRENT\_DATE - INTERVAL '1 month')

AND order\_date < DATE\_TRUNC('month', CURRENT\_DATE)

GROUP BY product\_id, tier

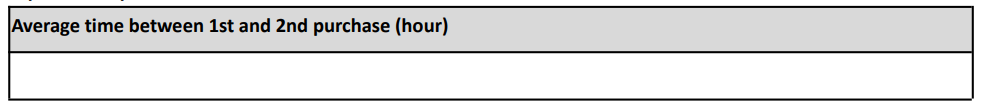
ORDER BY qty\_sold DESC;

**Q3:** The following buyers purchased on Shopee on separate days and on several occasions:

* 123456
* 987654
* 34567

Find the average time (in hrs) between their first and second checkout in the last 120 days.

**Expected output:**



**Solution:**

WITH buyer\_checkouts AS (

    SELECT

        buyer\_id,

        ROW\_NUMBER() OVER (PARTITION BY buyer\_id ORDER BY order\_date) AS checkout\_order,

        order\_date

    FROM order\_trans

    WHERE buyer\_id IN (123456, 987654, 34567)

    AND order\_date >= CURRENT\_DATE - INTERVAL '120 days'

)

SELECT

    buyer\_id,

    AVG(EXTRACT(EPOCH FROM (checkout\_2\_date - checkout\_1\_date))) / 3600 AS average\_time\_between\_checkouts\_hrs

FROM (

    SELECT

        buyer\_id,

        checkout\_order,

        order\_date AS checkout\_1\_date,

        LEAD(order\_date, 1) OVER (PARTITION BY buyer\_id ORDER BY order\_date) AS checkout\_2\_date

    FROM buyer\_checkouts

) AS checkout\_pairs

WHERE checkout\_order = 1

AND checkout\_2\_date IS NOT NULL

GROUP BY buyer\_id;

**Part 2: Writing Pseudocodes & Python Codes**

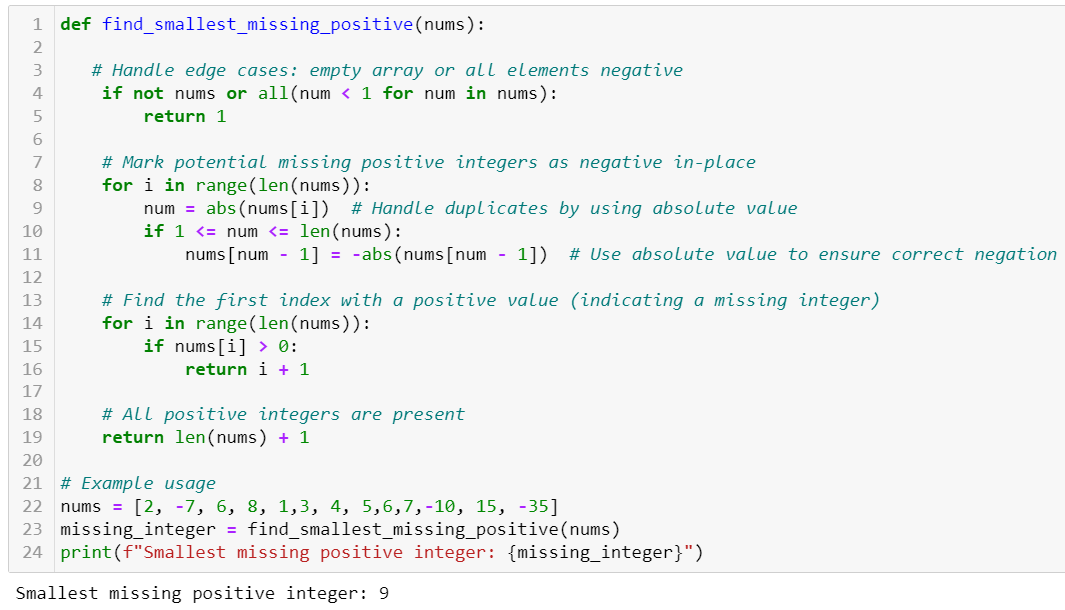
1. **Write a function that will be able to return the smallest positive integer missing from an unsorted array.**

***For example:***

**Input:** [2, 3, -7, 6, 8, 1, -10, 15, -35]

**Output:** 4

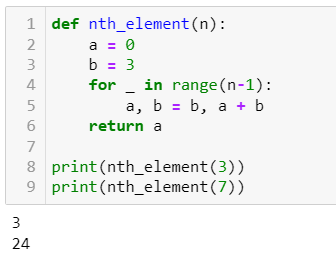
**Solution:**



1. **Write a function that will be able to return the Nth element from the following sequence: 0, 3, 3, 6, 9, 15, 24, 39, 63, 102, 165,...**

For e.g. if N = 3, output = 3 ; if N = 7, output = 24

**Solution:**



1. **Write a function that will be able to remove duplicates from a given array.**

Output requirements:

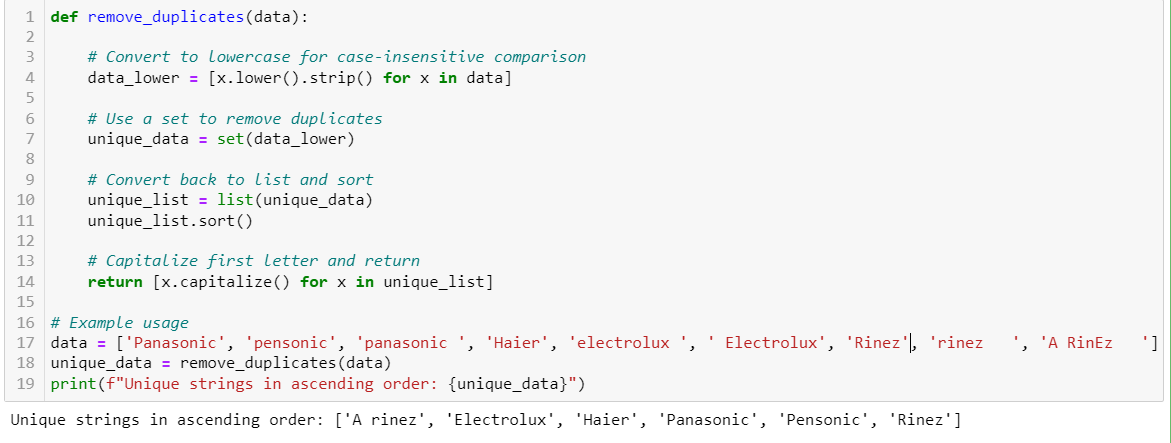
1. Array consists of a unique list of strings
2. Ascending order
3. First letter of each array element (string) capitalized
4. All trailing and leading spaces removed

***For example:***

**Input:** [‘Panasonic’, pensonic’, ‘panasonic ‘, ‘Haier’, ‘electrolux ’, ‘ Electrolux’]

**Output:** [‘Electrolux’, ‘Haier’, Panasonic’, ‘Pensonic’]

**Solution:**



**Part 3: Open-Ended Question**

How would you develop data mart and data flow for rider incentive which will be used for the rider to check the incentive on daily basis.

Suggestions/Tips:

1. A rider will have 4 factors: delivery point, pickup point, return point, and delivery successful point (% of monthly delivery successful)
2. Incentives will be calculated cumulatively based on the sum of 4 factors
3. Think about the daily and monthly maintenance (detail data and summary) and flexibility of monthly base rate changes
4. Keep your answers structured

**Solution:**  
**Data Mart and Data Flow for Rider Incentive**

Objective: Develop a data mart and data flow to manage rider incentives, allowing daily incentive checks and flexibility for base rate changes.

**Factors:**

* Delivery point: Number of successful deliveries (monthly)
* Pickup point: Number of pickups completed (monthly)
* Return point: Number of on-time returns (monthly)
* Delivery successful point: Percentage of successful deliveries from total deliveries (monthly)

**Incentive Calculation:**

* Cumulative: Sum of individual factor points weighted by their respective coefficients.
* Coefficients: Adjustable to prioritize specific factors.

**Data Mart and Flow:**

***1. Data Sources:***

* Operational Data Store (ODS): Real-time data on deliveries, pickups, returns, and success rates.
* Master Data: Rider information, base rates, coefficient values.

***2. Data Staging:***

* Extract data from ODS and Master Data at scheduled intervals (daily/hourly).
* Transform data into a consistent format:
  + Deduplicate entries.
  + Calculate monthly point values for each factor.
  + Apply coefficients to each factor.

***3. Data Mart:***

* Dimension tables: Rider, Time (date/month), Base Rate.
* Fact table: RiderIncentive (rider\_id, date, delivery\_points, pickup\_points, return\_points, delivery\_success\_points, total\_points, incentive\_amount).
* Aggregate data for daily checks: Calculate total points and corresponding incentive amount based on base rate and formula.
* Store monthly summaries for historical analysis.

***4. Data Flow:***

* ETL process extracts, transforms, and loads data from ODS and Master Data to the Data Mart.
* Scheduled jobs (daily/monthly) refresh data and recalculate incentives.
* API or UI interface allows riders to check their daily incentive based on ID and date.

***5. Maintenance and Flexibility:***

* Daily: Update Data Mart with new data, recalculate and store daily incentives.
* Monthly: Aggregate data for monthly summaries, update base rates if needed.
* Flexibility:
  + Adjustable coefficients to prioritize factors based on campaign goals.
  + Configurable base rate changes through Master Data updates.

***6. Structured Response:***

This structured response outlines the key components of the data mart and data flow for rider incentives:

* Data sources and their roles.
* Data staging and transformation process.
* Data Mart schema with dimensions and facts.
* Data flow and refresh schedule.
* Maintenance and flexibility considerations.

***Additional Tips:***

* Implement data quality checks to ensure accuracy and consistency.
* Consider caching frequently accessed data for performance optimization.
* Design dashboards and reports for incentive trends and analysis.
* Securely store and access sensitive rider data.

By implementing this data mart and data flow, you can provide riders with clear and accessible daily incentive information while maintaining flexibility for adjusting base rates and campaign priorities.