💻 Question 01: E-commerce Recommendation System (KarachiDeals)

Goal: Implement a "Customers Who Bought This Also Bought" feature by finding which products are most frequently purchased together. **Topics:** Data Structures (Lists, Dictionaries, Sets, Tuples), Functions, Loops, `zip`, `enumerate`.

Provided Data

In [1]:
```
transactionLog = [
    {'orderId': 1001, 'customerId': 'cust_Ahmed', 'productId': 'prod
_10'},
    {'orderId': 1001, 'customerId': 'cust_Ahmed', 'productId': 'prod
_12'},
    {'orderId': 1002, 'customerId': 'cust_Bisma', 'productId': 'prod
_10'},
    {'orderId': 1002, 'customerId': 'cust_Bisma', 'productId': 'prod
_15'},
    {'orderId': 1003, 'customerId': 'cust_Ahmed', 'productId': 'prod
_15'},
    {'orderId': 1004, 'customerId': 'cust_Faisal', 'productId': 'pro
d_12'},
    {'orderId': 1004, 'customerId': 'cust_Faisal', 'productId': 'pro
d_10'},
]

productCatalog = {
    'prod_10': 'Wireless Mouse',
    'prod_12': 'Keyboard',
    'prod_15': 'USB-C Hub',
}
```

Tasks

1. **Transform Data:** Write a function `processTransactions(transactionsList)` that returns a dictionary where keys are **customerIds** and values are a **set of their purchased productIds**.

In [2]:
```python
def processTransactions(transactionsList):
    """
    Transforms a list of transaction dictionaries into a dictionary where keys are
    customerIds and values are a set of their purchased productIds.
    """
    customer_data = {}
    for transaction in transactionsList:
        customerId = transaction['customerId']
        productId = transaction['productId']
        if customerId not in customer_data:
            customer_data[customerId] = set()
        customer_data[customerId].add(productId)
    return customer_data

# Example usage
customer_purchases = processTransactions(transactionLog)
print("Customer Purchase Data:")
print(customer_purchases)
```

```
Customer Purchase Data:
{'cust_Ahmed': {'prod_15', 'prod_10', 'prod_12'}, 'cust_Bisma': {'prod_15', 'prod_10'}, 'cust_Faisal': {'prod_10', 'prod_12'}}
```

1. **Find Pairs:** Write a function `findFrequentPairs(customerData)` to count how many times any two products were bought together.
    - Use a dictionary for tracking co-purchases.
    - The key for a pair must be immutable and order-independent (e.g., `('prod_10', 'prod_12')` is the same as `('prod_12', 'prod_10')`).

In [3]:
```
def findFrequentPairs(customerData):
    """
    Counts how many times any two products were bought together.

    Args:
        customerData: A dictionary where keys are customerIds and va
lues are
                      a set of their purchased productIds.

    Returns:
        A dictionary where keys are frozensets of two productIds and
values are
        the co-purchase count.
    """
    frequent_pairs = {}
    for products in customerData.values():
        product_list = sorted(list(products)) # Sort to ensure orde
r-independence
        for i in range(len(product_list)):
            for j in range(i + 1, len(product_list)):
                pair = frozenset({product_list[i], product_list[j]})
                if pair not in frequent_pairs:
                    frequent_pairs[pair] = 0
                frequent_pairs[pair] += 1
    return frequent_pairs

# Example usage
frequent_itemsets = findFrequentPairs(customer_purchases)
print("\nFrequent Item Pairs:")
print(frequent_itemsets)
```

```
Frequent Item Pairs:
{frozenset({'prod_10', 'prod_12'}): 2, frozenset({'prod_15', 'prod_1
0'}): 2, frozenset({'prod_15', 'prod_12'}): 1}
```

1. **Get Recommendation:** Write a function `getRecommendations(targetProductId, frequentPairs)` .
    - Iterate through the pairs map and find all pairs that include the `targetProductId` .
    - Return a **ranked list of the other products**, sorted by their co-purchase count (highest first).

In [4]:
```python
def getRecommendations(targetProductId, frequentPairs):
    """
    Finds all pairs that include the targetProductId and returns a ranked list
    of the other products, sorted by their co-purchase count (highest first).

    Args:
        targetProductId: The product ID for which to find recommendations.
        frequentPairs: A dictionary where keys are frozensets of two productIds
                       and values are the co-purchase count.

    Returns:
        A list of tuples, where each tuple contains (recommended_productId, count),
        sorted in descending order of count.
    """
    recommendations = []
    for pair, count in frequentPairs.items():
        if targetProductId in pair:
            # Find the other product in the pair
            other_product = list(pair - {targetProductId})[0]
            recommendations.append((other_product, count))

    # Sort recommendations by count in descending order
    recommendations.sort(key=lambda item: item[1], reverse=True)
    return recommendations

# Example usage
target_product = 'prod_10'
recommendations = getRecommendations(target_product, frequent_itemsets)
print(f"\nRecommendations for {target_product}:")
print(recommendations)
```

```
Recommendations for prod_10:
[('prod_12', 2), ('prod_15', 2)]
```

1. **Generate Report:** Write a function `generateReport(targetProductId, recommendations, catalog)` .
    - Use `zip` and `enumerate` to align the ranked recommendation data with their product names from the catalog.
    - Print a clean, **1-indexed report**.

In [5]:
```
def generateReport(targetProductId, recommendations, catalog):
    """
    Prints a 1-indexed report of recommendations, aligning recommend
ation data
    with product names from the catalog.

    Args:
        targetProductId: The product ID for which recommendations we
re generated.
        recommendations: A list of tuples (recommended_productId, co
unt).
        catalog: A dictionary mapping productIds to product names.
    """
    print(f"\nReport for recommendations for {catalog.get(targetProd
uctId, targetProductId)}:")
    print("----------------------------------------------------")
    if not recommendations:
        print("No recommendations found for this product.")
    else:
        for index, (recommended_productId, count) in enumerate(zip(r
ecommendations, [rec[1] for rec in recommendations])):
            product_name = catalog.get(recommended_productId[0], rec
ommended_productId[0])
            print(f"{index + 1}. {product_name} (Co-purchased {coun
t} times)")
    print("----------------------------------------------------")

# Example usage
target_product = 'prod_10'
recommendations = getRecommendations(target_product, frequent_itemse
ts)
generateReport(target_product, recommendations, productCatalog)
```

```
Report for recommendations for Wireless Mouse:
----------------------------------------------------
1. Keyboard (Co-purchased 2 times)
2. USB-C Hub (Co-purchased 2 times)
----------------------------------------------------
```

In [4]: