□ Low-Level Design (LLD) – Bank Transaction Analyzer

Difficulty Level: Easy | Total Marks: 20
Standards Followed: 4 Functions | 4 Visible Test Cases

□ Summary of Corrections (Based on SME Feedback)

- □ Validated input file structure with correct columns
- □ Used proper filtering and aggregation with Pandas
- □ Ensured each function is independently testable
- □ Return types and formats match test case structure

□ Concepts Tested
□ CSV Reading with Pandas
□ Filtering rows based on conditions
□ Grouping and summing transaction amounts
□ Sorting and selecting Top-N records

□ Problem Statement
You are provided with a CSV file that logs transactions for a bank. Each transaction includes the customer ID, transaction type (credit/debit), amount, and date.

Your task is to perform transaction-based analytics using Pandas.

□ Operations

□ 1. Load Transaction Data
□ Load the transaction CSV file into a DataFrame.

□ Function Prototype:

```
def load_transactions(file_path: str) -> pd.DataFrame:
```

□ Input: `"transactions.csv"`
□ Output: DataFrame

□ Implementation Flow:

- Use `pd.read_csv(file_path)`
- Return the DataFrame

□ **2. Total Amount by Transaction Type**
□ Calculate total credited and debited amounts.

□ Function Prototype:

```
def total_by_type(df: pd.DataFrame) -> dict:
```

□ Input: DataFrame
□ Output: Dictionary → {"credit": total_credit, "debit": total_debit}

□ Implementation Flow:

- Use filtering for type = credit/debit
- Sum amounts separately
- Return dictionary

□ **3. Get High Value Transactions**
□ Return all transactions above ₹10,000.

□ Function Prototype:

```
def high_value_transactions(df: pd.DataFrame) -> pd.DataFrame:
```

□ Input: DataFrame
□ Output: Filtered DataFrame

□ Implementation Flow:

- Filter rows where amount > 10000
- Return resulting DataFrame

□ **4. Top 3 Customers by Total Transaction Amount**
□ Identify top customers based on cumulative amount.

□ Function Prototype:

```
def top_customers(df: pd.DataFrame) -> list:
```

□ Input: DataFrame
□ Output: List of tuples → [(customer_id, total_amount), ...]

□ Implementation Flow:

- Group by customer_id
- Sum amounts
- Sort descending and get top 3
- Return as list of tuples

□ Implementation Hints

```
# Implementation stubs only
import pandas as pd

class BankTransactionAnalyzer:

    def load_transactions(self, file_path: str) -> pd.DataFrame:
        pass  # TODO

    def total_by_type(self, df: pd.DataFrame) -> dict:
        pass  # TODO

    def high_value_transactions(self, df: pd.DataFrame) -> pd.DataFrame:
        pass  # TODO

    def top_customers(self, df: pd.DataFrame) -> list:
        pass  # TODO
```

□ Test Cases & Marks Allocation

Test Case ID	Description	Associated Function	Marks
TC1	Load CSV into DataFrame	load_transactions()	□ 5
TC2	Total credit and debit	total_by_type()	□ 5
TC3	Get high value transactions	high_value_transactions()	□ 5
TC4	Get top 3 customers	top_customers()	□ 5
	Total Marks	–	□ 20

□ Visible Test Cases

□ TC1: Load CSV

```
Input: "transactions.csv"
Expected Output: Valid DataFrame with columns ["customer_id", "type",
"amount", "date"]
```

□ TC2: Transaction Totals

```
df = load_transactions("transactions.csv")
total_by_type(df)
Expected Output: {"credit": 150000.0, "debit": 90000.0}
```

□ TC3: High Value Filter

```
df = load_transactions("transactions.csv")
high_value_transactions(df)
Expected Output: DataFrame with 5 rows (amount > 10000)
```

□ TC4: Top Customers

```
df = load_transactions("transactions.csv")
top_customers(df)
Expected Output: [(101, 50000), (205, 40000), (112, 35000)]
```