Design Decisions:

In my design, I have crafted the IntHistogram class to create histograms for integer values, aiding in database query optimization. This class initializes a histogram with a defined number of buckets and a range of values. The histogram's buckets are populated by incrementing counts in the appropriate bucket as values are added through the addValue method. The histogram's primary function is to facilitate query optimization by estimating the selectivity of various predicate operations, such as EQUALS, GREATER_THAN, and LESS_THAN, through the estimateSelectivity method. This method calculates the likelihood of a value fulfilling a particular predicate by examining its frequency within the histogram's buckets, adjusting for out-of-range inputs.

Simultaneously, I have also implemented the JoinOptimizer class, aimed at enhancing join operations in databases. This class includes two key methods: estimateJoinCost and estimateTableJoinCardinality. The former calculates the cost of a join operation based on the estimated cardinalities and costs of scanning each table involved in the join. The latter method estimates the cardinality of the join result, considering factors like whether the joining fields are primary keys. The most complex method, orderJoins, systematically evaluates different combinations of join orders. It uses a dynamic programming approach to identify the most cost-effective sequence of joins, considering various subsets of joins and computing the cost and cardinality for each subset. My implementation extensively utilizes the C++ Standard Library for efficient data manipulation and management, incorporating vectors, unordered maps, and sets.

API Modifications:

Upon thorough evaluation of the provided Application Programming Interface (API), we determined that no modifications or adjustments were essential for our implementation's objectives. Thus, the API remains unaltered.

Code Completeness:

As per the stipulated requirements for PA4, we have successfully realized and executed all essential components, ensuring a holistic solution.

Implementation Duration and Challenges:

The entire duration of PA4's implementation spanned approximately one full week. Throughout this period, we diligently committed changes to our GitHub repository over 30 times, ensuring version control best practices and tracking our progress.

Collaboration Details:

Our team employed a collaborative approach to tackle the project. Specifically:

Yanpeng Zhao spearheaded the tasks encompassed in EX4.1
Pengfei Li undertook the challenges presented in EX4.2

While each member primarily focused on their designated segments, our collaboration wasn't confined strictly to these demarcations. We embraced collective brainstorming, debugging, and testing, ensuring that the final output was a product of combined expertise and team synergy.