**Work through the following materials this week, making sure that you can do the given activities.**

1. **Chapter 17**
   1. **B+ Trees — Focus on the Section 17.3 (skipping the material on algorithms).**
      * **Compare and contrast *B-trees* and *B+-trees*.**B-tree has additional constraints that ensure that the tree is always balanced. Insertion and deletion becomes more complicated.  
        B+- trees similar to a B-tree but the data pointers are stored only at the leaf nodes of the tree.
      * **Explain what it means for such a tree to be *balanced*.**All the leaf nodes are on the same level.
      * **Compare and contrast *tree* and *data* pointers.  
        Tree pointer-** a pointer to another node in the B-tree.  
        **Data pointer-** a pointer to the record whose search key field value is equal to Ki(data file block containing that record).
   2. **Explain the nature of *multiple-attribute indexes* (see the introduction to Section 17.4).**multiple-attribute indexes are for indexing multiple columns in order to increase the speed of getting information that is grouped together. This only increase the speed if something is grouped up in the thing it is ordered by. For example finding someone using their last name, then first name is faster than the opposite if the database orders it by last name.
   3. **Explain the nature of *hash* and *bitmap* indexes (see Sections 17.5.1–2).  
      Hash index:** Secondary structure to access the file by using hashing on a search key.  **Bitmap index:** creates an index for one or more columns, and each value or value range in those columns. Mostly used for relations that contain a large number or rows.
   4. **Physical Database Design — Focus on the Section 17.7.**
      * **Explain the concept of *physical database design* (PDD).**Goal is to create appropriate structuring of data and with good performance. Designers must know what queries and data needs to be stored and run tests to see how fast the data base is and consider everything.
      * **Compare and contrast the uses for B-tree, hash and bitmap indexes in PDD.**In general RDBMSs use B+-trees for indexing. Hash indexes work well with joins to find matching records but they do not support range queries.
      * **Explain whether *denormalization* should be considered a form of PDD.**Denormalization is not a form of pdd. ?? I am not sure about this question.
2. **Chapter 18**
   1. **Explain the concept of *query optimization* (see the introduction to the chapter).**Process for choosing a suitable execution strategy for a query.
3. **Oracle’s EXPLAIN PLAN**
   1. **Load the small version of the Internet Movie DB defined in the repo (cs342/databases/imdb), configure the auto-trace facility in SQL\*Plus (SET SERVEROUTPUT ON; & SET AUTOTRACE ON;) and generate an execution plan for some simple query. What is the execution plan operation used and what does that operation do?**
   2. **Explain at least the following operations (see** [**Oracle Execution Plan Operations**](http://use-the-index-luke.com/sql/explain-plan/oracle/operations)**).**
      * **TABLE ACCESS FULL -** expensive. Goes through all rows and columns.
      * **TABLE ACCESS BY INDEX -** retrieves row from table using rowId.
      * **NESTED LOOPS -** joins two tables by fetching the result from one table and querying the other table for each row from the first.
      * **HASH JOIN -** Joins candidate records from one side of the join into a hash table that is then probed for each row from the other side of the join.
      * **INDEX UNIQUE SCAN -** The INDEX UNIQUE SCAN performs the B-tree traversal only. The database uses this operation if a unique constraint ensures that the search criteria will match no more than one entry. See also Chapter 1, “Anatomy of an SQL Index”.