

Pune Institute of Computer Technology, Pune

Department of Computer Engineering

A.Y. 2020-21 Semester: I

Database Management System Lab

Name: Aditya Sawant

Roll No: 31302

Batch: K3

ASSIGNMENT 2

TITLE:

Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as creation of (Table, View, Index, Sequence, Synonym).

PROBLEM DEFINITION:

Implement DDL commands in context of view, index, sequence using JDBC.

OBJECTIVE:

- To understand implementation of JDBC.
- To understand & implement the various DDL and DML Commands.
- To understand database concepts like view, index ,sequence and synonym.

OUTCOME:

We will be able to understand and implement DDL and DML commands using JDBC.

HARDWARE REQUIREMENTS:

- MONITOR
- KEYBOARD
- 2GB RAM
- 2.4GHz I5 PROCESSOR

SOFTWARE REQUIREMENTS:

- DATABASE-MYSQL
- IDE-ECLIPSE
- OS-FEDORA 20

Theory:

JDBC:

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation.

STEPS INVOLVED IN JDBC:

The fundamental steps involved in the process of connecting to a database and executing a query consist of the following:

1. Import JDBC packages.
2. Load and register the JDBC driver.
3. Open a connection to the database.
4. Create a statement object to perform a query.
5. Execute the statement object and return a query resultset.
6. Process the resultset.
7. Close the resultset and statement objects.
8. Close the connection.

Applications of JDBC:

Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as –

- Java Applications
- Java Applets
- Java Servlets
- Java ServerPages (JSPs)
- Enterprise JavaBeans (EJBs).

INDEX:

An index can be created in a table to find data more quickly and efficiently. The users cannot see the indexes, they are just used to speed up searches/queries.

SYNTAX-

```
CREATE INDEX index_name ON table_name (column_name);
```

SEQUENCE:

A sequence is a set of integers 1, 2, 3, ... that are generated in order on a specific demand. Sequences are frequently used in the databases because many applications require each row in a table to contain a unique value and sequences provide an easy way to generate them.

The simplest way in MySQL to use Sequences is to define a column as AUTO_INCREMENT and leave the remaining things to MySQL to take care.

SYNTAX-

```
CREATE Sequence sequence-name start with initial-value increment by increment-value maxvalue  
maximum-value cycle|nocycle
```

SYNONYM:

Synonyms provide both data independence and location transparency. Synonyms permit applications to function without modification regardless of which user owns the table or view and regardless of which database holds the table or view. However, synonyms are not a substitute for privileges on database objects. Appropriate privileges must be granted to a user before the user can use the synonym.

SYNTAX-

Use the CREATE SYNONYM statement to create a synonym

Code:

```
DEMO - JDBC/Java/MyClass.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

MyClass.java II
1 import java.sql.*;
2 import java.util.Scanner;
3 public class MyClass {
4 {
5     Connection con;
6     Statement st;
7     ResultSet rs;
8     MyClass() {
9     {
10         try {
11             Class.forName("com.mysql.jdbc.Driver");
12             con = DriverManager.getConnection("jdbc:mysql://localhost/professor", "root", "root");
13             st = con.createStatement();
14         } catch (Exception e) {
15             // TODO Auto-generated catch block
16             e.printStackTrace();
17         }
18     }
19 }
20 void display(String tableName) {
21     try {
22         while(rs.next())
23         {
24             if(tableName.equals("Departments"))
25             {
26                 let dept_id = rs.getInt(1);
27                 String dept_name = rs.getString(2);
28                 System.out.println(dept_id + " " + dept_name);
29             }
30             else if(tableName.equals("Professors"))
31             {
32                 let prof_id = rs.getInt(1);
33                 String prof_fname = rs.getString(2);
34                 String prof_lname = rs.getString(3);
35                 let dept_id = rs.getInt(4);
36                 String designation = rs.getString(5);
37                 let salary = rs.getInt(6);
38                 String doj = rs.getString(7);
39                 String email = rs.getString(8);
40             }
41         }
42     }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }
```

```
DEMO - JDBC/Java/MyClass.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

MyClass.java II
200 void display(String tableName) {
201     try {
202         while(rs.next())
203         {
204             if(tableName.equals("Departments"))
205             {
206                 let dept_id = rs.getInt(1);
207                 String dept_name = rs.getString(2);
208                 System.out.println(dept_id + " " + dept_name);
209             }
210             else if(tableName.equals("Professors"))
211             {
212                 let prof_id = rs.getInt(1);
213                 String prof_fname = rs.getString(2);
214                 String prof_lname = rs.getString(3);
215                 let dept_id = rs.getInt(4);
216                 String designation = rs.getString(5);
217                 let salary = rs.getInt(6);
218                 String doj = rs.getString(7);
219                 String email = rs.getString(8);
220                 long phone = rs.getLong(9);
221                 String city = rs.getString(10);
222                 System.out.println(prof_id + " " + prof_fname + " " + prof_lname + " " + dept_id + " " + designation + " " + salary + " " + doj + " " + email + " " + phone);
223             }
224             else if(tableName.equals("Works"))
225             {
226                 let prof_id = rs.getInt(1);
227                 let dept_id = rs.getInt(2);
228                 let duration = rs.getInt(3);
229                 System.out.println(prof_id + " " + dept_id + " " + duration);
230             }
231             else
232             {
233                 let prof_id = rs.getInt(1);
234                 String shift = rs.getString(2);
235                 let working_hours = rs.getInt(3);
236                 System.out.println(prof_id + " " + shift + " " + working_hours);
237             }
238         }
239     }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }
```

```

DEMS - JDBC/Java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

MyClass.java II
476 void add(String tableName)
477 {
478     Scanner sc = new Scanner(System.in);
479     String query = "";
480     if(tableName.equals("Departments"))
481     {
482         int dept_id = sc.nextInt();
483         sc.nextLine();
484         String dept_name = sc.nextLine();
485         query = "INSERT INTO departments " + "VALUES (" + dept_id + "," + dept_name + ")";
486     }
487     else if(tableName.equals("Professors"))
488     {
489         int prof_id = sc.nextInt();
490         sc.nextLine();
491         String prof_name = sc.nextLine();
492         String prof_lname = sc.nextLine();
493         int dept_id = sc.nextInt();
494         sc.nextLine();
495         String designation = sc.nextLine();
496         int salary = sc.nextInt();
497         sc.nextLine();
498         String dept = sc.nextLine();
499         String email = sc.nextLine();
500         long phone = sc.nextLong();
501         sc.nextLine();
502         String city = sc.nextLine();
503         query = "INSERT INTO professors " + "VALUES (" + prof_id + "," + prof_name + "," + prof_lname + "," + dept_id + "," + designation + "," + salary + "," + dept + "," + phone + "," + city + ")";
504     }
505     else if(tableName.equals("works"))
506     {
507         int prof_id = sc.nextInt();
508         int dept_id = sc.nextInt();
509         int duration = sc.nextInt();
510         query = "INSERT INTO works " + "VALUES (" + prof_id + "," + dept_id + "," + duration + ")";
511     }
512     else
513     {
514         int prof_id = sc.nextInt();
515     }
516 }

```

```

DEMS - JDBC/Java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

MyClass.java II
118 try
119 {
120     st.executeUpdate(query);
121 }
122 catch(Exception e) {
123     System.out.println(e);
124 }
125 }
126 }
127 void createTableDepartments() throws SQLException
128 {
129     String query = "create table departments( "
130     + "dept_id integer(20) primary key auto_increment, "
131     + "dept_name varchar(20))";
132     st.executeUpdate(query);
133 }
134 void createTableProfessors() throws SQLException
135 {
136     String query = "create table professors( "
137     + "prof_id integer(20) primary key auto_increment, "
138     + "prof_name varchar(30), "
139     + "prof_lname varchar(30), "
140     + "dept_id integer(20) not null, "
141     + "designation varchar(15), "
142     + "salary int(10), "
143     + "dept varchar(20), "
144     + "email varchar(30), "
145     + "phone int(10), "
146     + "city varchar(15), "
147     + "foreign key (dept_id) "
148     + "references departments(dept_id) on delete cascade); ";
149     st.executeUpdate(query);
150 }
151 void createTableworks() throws SQLException
152 {
153     String query = "create table works( "
154     + "prof_id integer(20) not null, "

```

```

DEMS - JDBC/MyClass.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

MyClass.java
142 st.executeUpdate(query);
143 }
144
145 void createTablemarks() throws SQLException
146 {
147     String query = "create table worksi "
148     + "prof_id integer(20) not null , "
149     + "dept_id integer(20) not null , "
150     + "duration integer(5) , "
151     + "foreign key (dept_id) "
152     + "references departments(dept_id) on delete cascade,"
153     + "foreign key (prof_id) "
154     + "references professors(prof_id) on delete cascade;";
155     st.executeUpdate(query);
156 }
157
158 void createTableShift() throws SQLException
159 {
160     String query = "create table shift( "
161     + "prof_id integer(20) not null , "
162     + "shift integer(5) not null , "
163     + "working_hours integer(5) not null , "
164     + "primary key (prof_id,shift,working_hours);";
165     st.executeUpdate(query);
166 }
167
168 void query1() throws SQLException
169 {
170     System.out.println("1. Display all customer details with city pune and mumbai and customer first name starting with 'a' or 'b'.");
171     String query = "select * from professors where (city = 'Pune' or city = 'Mumbai') and (prof_name like 'a%' or prof_name like 'b%')";
172     rs = st.executeQuery(query);
173     display("Professors");
174 }
175
176 void query2() throws SQLException
177 {
178     System.out.println("2. List the number of different customer cities.");
179     String query = "select count(distinct city) as total from professors;";
180 }

```

Outputs:

```

DEMS - JDBC/MyClass.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

MyClass [Java Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (25 Aug 2020 10:38:50 AM)

1
2 Mechanical
3 Electrical
4 Electronics
5 Instrumental
6 Computer Science
7 IT
8 Chemical
9 Astronomy
10 Aviation
11 Law
12 Kunal Sharma 2 HOD 150000 2000-04-15 ksharma@gmail.com 784512781 Pune
13 Suraj Singh 4 Professor 100000 2017-05-12 ssingh@gmail.com 90552252 Mumbai
14 Manjit Patel 7 Asst Professor 50000 2018-11-02 rpatel@gmail.com 895423834 Shanghai
15 Dharmash Singh 4 Asst Professor 60000 2015-07-14 dsingh@gmail.com 741862052 Pune
16 Anant Khat 1 Professor 110000 2011-05-14 akhat@gmail.com 903214703 Mumbai
17 Vivek Chandra 4 HOD 200000 2001-04-11 schandra@gmail.com 875470129 Thane
18 Raju Deshmukh 3 HOD 100000 2004-07-17 rdeshmukh@gmail.com 651270304 Mumbai
19 Divya Shukla 1 Asst Professor 90000 2010-07-07 dsukla@gmail.com 987832145 Mumbai
20 Siddhant Patil 2 Professor 40000 2016-01-01 spatil@gmail.com 481270349 Pune
21 Varun Sharma 5 Asst Professor 10000 2015-01-01 vsharma@gmail.com 794612052 Mumbai
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

The screenshot shows a Java IDE with a menu-driven application. The menu options are: 1. Create tables, 2. Enter data, 3. Display tables, 4. Show view, 5. Show index, 6. Exit. The application is running, and the output shows a list of 10 employees with their details: 1. Ravi Sharma, 2. Sunny Singh, 3. Ravijit Patel, 4. Dharmesh Singh, 5. Anant Bhat, 6. Shreyas Chopra, 7. Raju Deshmukh, 8. Dhruv Shukla, 9. Siddhant Patil, 10. Varun Sharma. The application is running on a Windows 10 desktop.

The screenshot shows a Java IDE with a menu-driven application. The menu options are: 1. Create tables, 2. Enter data, 3. Display tables, 4. Show view, 5. Show index, 6. Exit. The application is running, and the output shows a list of 10 employees with their details: 1. Ravi Sharma, 2. Sunny Singh, 3. Ravijit Patel, 4. Dharmesh Singh, 5. Anant Bhat, 6. Shreyas Chopra, 7. Raju Deshmukh, 8. Dhruv Shukla, 9. Siddhant Patil, 10. Varun Sharma. The application is running on a Windows 10 desktop.

CONCLUSION:

WE HAVE SUCCESSFULLY UNDERSTOOD CONCEPTS OF VIEW, INDEXING , SEQUENCE AND SYNONYM AND IMPLEMENTED DDL AND DML COMMANDS USING JDBC.