Faculty of Engineering and Technology
Bachelor of Technology
Effective from Academic Year 2019-20

Subject Code: 1ET1030404	**Subject Title: Database Management Systems**
Pre-requisite	---

Course Objective: The core objective of this course is to make the students aware and have some expertise with various facets of database design methodology in oracle database, Microsoft SQL Server, MySQL etc. used in industry now days. This course aims to present various, Relational Database, Storage Architecture, Query Language & Processing, Normalization, Function Dependency used for database design during software development. This course shall act as platforms for the students wishing to carry out further development in database management for large enterprise.

Teaching Scheme (Hours per week)				**Evaluation Scheme (Marks)**				
				Theory		**Practical**		
Lecture	**Tutorial**	**Practical**	**Credit**	**University Assessment**	**Continuous Assessment**	**University Assessment**	**Continuous Assessment**	**Total**
3	-	2	4	70	30	30	20	150

Subject Contents			
Sr. No	**Topic**	**Total Hours**	**Weight (%)**
1.	**Introduction:** Purpose & Applications, Database Architecture, Type of DBMS, Data Model, Users Role & Responsibility, Transaction Management	02	05
2.	**Relational Model:** Structure of Relational Database, Database Schema, Keys, Relational Algebra Operations, Relational Algebra Query Language	04	10
3.	**SQL:** Overview of the SQL Query Language, SQL Data Definition, Basic Structure of SQL Queries, Integrity Constraint, Additional Basic Operations, Set Operations, Null Values, Aggregate Functions, Nested Sub queries, Modification of the Database, Join Expressions, Aggregation Features	04	10
4.	**E-R Model:** Overview, Constraints, E-R Diagrams, E-R Design Issues, Advance E-R Model	03	10
5.	**Database Design:** Features of Good Database Design, Redundancy, Functional Dependency, Decomposition, Normalizations, Normal Forms,	05	15
6.	**Query Processing & Optimization:** Overview, Estimating of Query Cost, Evaluation of Expressions, Evaluation Plan, Selection Operation, Sorting, Join Operations, Indexing, Ordered Indices, B+ Tree Index, Static and Dynamic Hashing	05	15
7.	**Transaction Management:** Concept, Transaction Model, ACID Property, Serializability, Transaction Protocols, Deadlock Handling, Recovery	04	15
8.	**Advance SQL:** Views, Index, Transactions, Authorization, Functions and Procedures, Triggers, Cursor	02	10
9.	**Case Study:** Postgre SQL, MySQL	02	10

Course Outcome:
After learning the course, the students should be able to:

- Understand the business data and relation between data
- Use of Technology to store the large data
- Efficiently usage of data in short response time
- Access control over the data
- Design of database for real life project

List of References:

1. Database System Concepts, Abraham Silberschatz, Henry F. Korth & S. Sudarshan, McGraw Hill.
2. An introduction to Database Systems, C J Date, Addition-Wesley.
3. SQL- PL/SQL by Ivan bayross
4. Oracle – The complete reference – TMH /oracle press

E-Resources / Web Links

1. www.codex.cs.yale.edu/avi/db-book/
2. www.pages.cs.wisc.edu/~dbbook/
3. www.techtud.com/computer-science-and-information-technology/database-management-system
4. www.techtud.com/course/practical-approach-dbms
5. www.dadbm.com/dbms-blog/
6. www.studytonight.com/dbms/

List of Experiments:
Note: The experiment list provided beneath is for reference only. The course teacher may change/formulate it as per his/her methodology and requirement.

Prepare Database

Table Name: ClassRoom		
Column Name	**Data Type**	**Constraint**
building	varchar(15)	
room_number	varchar(7)	Primary Key
capacity	numeric(4,0)	

building	**room_number**	**capacity**
Packard	101	500
Painter	514	10
Taylor	3128	70
Watson	100	30
Watson	120	50

Table Name: Department		
Column Name	**Data Type**	**Constraint**
dept_name	varchar(20)	Primary Key
building	varchar(15)	
budget	numeric(12,2)	budget > 0

dept_name	**building**	**budget**
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Table Name: Course		
Column Name	**Data Type**	**Constraint**
course_id	varchar(8)	Primary Key
title	varchar(50)	
dept_name	varchar(20)	Foreign Key (Department)
credits	numeric(2,0)	credits > 0

course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Table Name: Instructor		
Column Name	**Data Type**	**Constraint**
Instructor_ID	varchar(5)	Primary Key
name	varchar(20)	Not Null
dept_name	varchar(20)	Foreign Key (Department)
salary	numeric(8,2)	salary > 29000

Instructor_ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Table Name: Section		
Column Name	**Data Type**	**Constraint**
course_id	varchar(8)	Foreign Key (Course)
sec_id	varchar(8)	
semester	varchar(6)	semester in ('Fall', 'Winter', 'Spring', 'Summer')
year	numeric(4,0)	year > 1701 and year < 2100
building	varchar(15)	Foreign Key (ClassRoom)
room_number	varchar(7)	
time_slot_id	varchar(4)	Foreign Key (time_slot)

Composite Primary key (course_id, sec_id, semester, year)

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

Table Name: Teaches		
Column Name	**Data Type**	**Constraint**
Instructor_ID	varchar(5)	Foreign Key (Instructor)
course_id	varchar(8)	Foreign key (Section)
sec_id	varchar(8)	
Semester	varchar(6)	
year	numeric(4,0)	
Composite Primary key (Instructor_ID, course_id, sec_id, semester, year)		

Instructor_ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

SANKALCHAND PATEL UNIVERSITY
।। अथातो ज्ञानजिज्ञासा ।।

Faculty of Engineering and Technology
Second Year (4th Sem) Bachelor of Technology
In Effect from Academic Year 2017-18

Table Name: Student		
Column Name	**Data Type**	**Constraint**
Student_ID	varchar(5)	Primary Key
name	varchar(20)	Not Null
dept_name	varchar(20)	Foreign key (Department)
tot_cred	numeric(3,0)	tot_cred >= 0

Student_ID	**name**	**dept_name**	**tot_cred**
128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

Table Name: Takes		
Column Name	**Data Type**	**Constraint**
Student_ID	varchar(5)	Foreign key (Student)
course_id	varchar(8)	Foreign key (Section)
sec_id	varchar(8)	
Semester	varchar(6)	
Year	numeric(4, 0)	
grade	varchar(2)	
Composite Primary key (Student_ID, course_id, sec_id, semester, year)		

Student_ID	**course_id**	**sec_id**	**semester**	**year**	**grade**
128	CS-101	1	Fall	2009	A
128	CS-347	1	Fall	2009	A-
12345	CS-101	1	Fall	2009	C
12345	CS-190	2	Spring	2009	A
12345	CS-315	1	Spring	2010	A
12345	CS-347	1	Fall	2009	A
19991	HIS-351	1	Spring	2010	B
23121	FIN-201	1	Spring	2010	C+
44553	PHY-101	1	Fall	2009	B-
45678	CS-101	1	Fall	2009	F
45678	CS-101	1	Spring	2010	B+
45678	CS-319	1	Spring	2010	B
54321	CS-101	1	Fall	2009	A-
54321	CS-190	2	Spring	2009	B+
55739	MU-199	1	Spring	2010	A-
76543	CS-101	1	Fall	2009	A
76543	CS-319	2	Spring	2010	A
76653	EE-181	1	Spring	2009	C
98765	CS-101	1	Fall	2009	C-
98765	CS-315	1	Spring	2010	B
98988	BIO-101	1	Summer	2009	A
98988	BIO-301	1	Summer	2010	NULL

Table Name: Advisor		
Column Name	**Data Type**	**Constraint**
Student_ID	varchar(5)	Primary key, Foreign key (Student)
Instructor_ID	varchar(5)	Foreign key (Instructor)

Student_ID	Instructor_ID
128	45565
12345	10101
23121	76543
44553	22222
45678	22222
76543	45565
76653	98345
98765	98345
98988	76766

Table Name: time_slot		
Column Name	**Data Type**	**Constraint**
time_slot_id	varchar(4)	Primary Key
day	varchar(1)	
start_hr	numeric(2)	0 <= start_hr < 24
start_min	numeric(2)	0 <= start_min < 60
end_hr	numeric(2)	0 <= end_hr < 24
end_min	numeric(2)	0 <= end_min < 60

time_slot_id	day	start_hr	start_min	end_hr	end_min
A	F	8	0	8	50
A	M	8	0	8	50
A	W	8	0	8	50
B	F	9	0	9	50
B	M	9	0	9	50
B	W	9	0	9	50
C	F	11	0	11	50
C	M	11	0	11	50
C	W	11	0	11	50
D	F	13	0	13	50
D	M	13	0	13	50
D	W	13	0	13	50
E	R	10	30	11	45
E	T	10	30	11	45
F	R	14	30	15	45
F	T	14	30	15	45
G	F	16	0	16	50
G	M	16	0	16	50
G	W	16	0	16	50
H	W	10	0	12	30

Table Name: prereq		
Column Name	**Data Type**	**Constraint**
course_id	varchar(8)	Foreign key (Course)
prereq_id	varchar(8)	Foreign key (Course)
Composite Primary key (course_id, prereq_id)		

course_id	prereq_id
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

Basic SQL

1. Find the names of all the instructors from Biology department
2. Find the names of courses in Computer science department which have 3 credits
3. For the student with ID 12345 (or any other value), show all course_id and title of all courses registered for by the student.
4. As above, but show the total number of credits for such courses (taken by that student). Don't display the tot_creds value from the student table, you should use SQL aggregation on courses taken by the student.
5. As above, but display the total credits for each of the students, along with the ID of the student; don't bother about the name of the student. (Don't bother about students who have not registered for any course, they can be omitted)
6. Find the names of all students who have taken any Comp. Sci. course ever (there should be no duplicate names)
7. Display the IDs of all instructors who have never taught a couse (Notesad1) Oracle uses the keyword minus in place of except; (2) interpret "taught" as "taught or is scheduled to teach")
8. As above, but display the names of the instructors also, not just the IDs.
9. You need to create a movie database. Create three tables, one for actors(AID, name), one for movies(MID, title) and one for actor_role(MID, AID, rolename). Use appropriate data types for each of the attributes, and add appropriate primary/foreign key constraints.
10. Insert data to the above tables (approx 3 to 6 rows in each table), including data for actor "Charlie Chaplin", and for yourself (using your roll number as ID).
11. Write a query to list all movies in which actor "Charlie Chaplin" has acted, along with the number of roles he had in that movie.
12. Write a query to list all actors who have not acted in any movie
13. List names of actors, along with titles of movies they have acted in. If they have not acted in any movie, show the movie title as null. (Do not use SQL outerjoin syntax here, write it from scratch.)

Intermediate SQL

Using the university schema, write the following queries. In some cases you need to insert extra data to show the effect of a particular feature -- this is indicated with the question. You should then show not only the query, but also the insert statements to add the required extra data.

1. Find the maximum and minimum enrollment across all sections, considering only sections that had some enrollment, don't worry about those that had no students taking that section
2. Find all sections that had the maximum enrollment (along with the enrollment), using a subquery.
3. As in in Q1, but now also include sections with no students taking them; the enrollment for such sections should be treated as 0. Do this in two different ways (and create require data for testing)
    a) Using a scalar subquery
    b) Using aggregation on a left outer join (use the SQL natural left outer join syntax)
4. Find all courses whose identifier starts with the string "CS-1"
5. Find instructors who have taught all the above courses
    a) Using the "not exists ... except ..." structure
    b) Using matching of counts which we covered in class (don't forget the distinct clause!).
6. Insert each instructor as a student, with tot_creds = 0, in the same department
7. Now delete all the newly added "students" above (note: already existing students who happened to have tot_creds = 0 should not get deleted)

8. Some of you may have noticed that the tot_creds value for students did not match the credits from courses they have taken. Write and execute query to update tot_creds based on the credits passed, to bring the database back to consistency. (This query is provided in the book/slides.)
9. Update the salary of each instructor to 10000 times the number of course sections they have taught.
10. Create your own query: define what you want to do in English, then write the query in SQL. Make it as difficult as you wish, the harder the better.

Advanced SQL

In this assignment, you will write more complex SQL queries, using the usual schema by default. Again, you have to add required data to test your queries, and must include the SQL statements to create the data along with your queries.

1. The university rules allow an F grade to be overridden by any pass grade (A, B, C, D). Now, create a view that lists information about all fail grades that have not been overridden (the view should contain all attributes from the takes relation).
2. Find all students who have 2 or more non-overridden F grades as per the takes relation, and list them along with the F grades.
3. Grades are mapped to a grade point as follows: A:10, B:8, C:6, D:4 and F:0. Create a table to store these mappings, and write a query to find the CPI of each student, using this table. Make sure students who have not got a non-null grade in any course are displayed with a CPI of null.
4. Find all rooms that have been assigned to more than one section at the same time. Display the rooms along with the assigned sections; I suggest you use a with clause or a view to simplify this query.
5. Create a view faculty showing only the ID, name, and department of instructors.
6. Create a view CSinstructors, showing all information about instructors from the Comp. Sci. department.
7. Insert appropriate tuple into each of the views faculty and CSinstructors, to see what updates your database allows on views; explain what happens.
8. Grant permission to one of your friends to view all data in your student relation.
9. Now grant permission to all users to see all data in your faculty view. Conversely, find a friend who has granted you permission on their faculty view, and execute a select query on that view.