Subject : 2305CS101 – Database Management Systems

Sr.	Practical
1	Database Name: **Branch_DIV_Rollno (Example: CSE_3C_101)** **Part A:** **Create following tables under above database.**

DEPOSIT

Column_Name	DataType
ACTNO	INT
CNAME	VARCHAR(50)
BNAME	VARCHAR(50)
AMOUNT	DECIMAL(8,2)
ADATE	DATETIME

BRANCH

Column_Name	DataType
BNAME	VARCHAR(50)
CITY	VARCHAR(50)

CUSTOMERS

Column_Name	DataType
CNAME	VARCHAR(50)
CITY	VARCHAR(50)

BORROW

Column_Name	DataType
LOANNO	INT
CNAME	VARCHAR(50)
BNAME	VARCHAR(50)
AMOUNT	DECIMAL(8,2)

Insert the data into above tables as shown below.

DEPOSIT

ACTNO	CNAME	BNAME	AMOUNT	ADATE
101	ANIL	VRCE	1000.00	1-3-95
102	SUNIL	AJNI	5000.00	4-1-96
103	MEHUL	KAROLBAGH	3500.00	17-11-95
104	MADHURI	CHANDI	1200.00	17-12-95
105	PRMOD	M.G. ROAD	3000.00	27-3-96
106	SANDIP	ANDHERI	2000.00	31-3-96
107	SHIVANI	VIRAR	1000.00	5-9-95
108	KRANTI	NEHRU PLACE	5000.00	2-7-95
109	MINU	POWAI	7000.00	10-8-95

BRANCH

BNAME	CITY
VRCE	NAGPUR
AJNI	NAGPUR
KAROLBAGH	DELHI
CHANDI	DELHI
DHARAMPETH	NAGPUR
M.G. ROAD	BANGLORE
ANDHERI	BOMBAY

VIRAR	BOMBAY
NEHRU PLACE	DELHI
POWAI	BOMBAY

CUSTOMERS

CNAME	CITY
ANIL	CALCUTTA
SUNIL	DELHI
MEHUL	BARODA
MANDAR	PATNA
MADHURI	NAGPUR
PRAMOD	NAGPUR
SANDIP	SURAT
SHIVANI	BOMBAY
KRANTI	BOMBAY
NAREN	BOMBAY

BORROW

LOANNO	CNAME	BNAME	AMOUNT
201	ANIL	VRCE	1000.00
206	MEHUL	AJNI	5000.00
311	SUNIL	DHARAMPETH	3000.00
321	MADHURI	ANDHERI	2000.00
375	PRMOD	VIRAR	8000.00
481	KRANTI	NEHRU PLACE	3000.00

From the above given tables perform the following SQL queries (SELECT Operation):

1. Retrieve all data from table DEPOSIT.
2. Retrieve all data from table BORROW.
3. Retrieve all data from table CUSTOMERS.
4. Insert a record (550,'JAY','AJNI',NULL)in the BORROW table.
5. Display Account No, Customer Name & Amount from DEPOSIT.
6. Display Loan No, Amount from BORROW.
7. Display loan details of all customers who belongs to 'ANDHERI' branch.
8. Give account no and amount of depositor, whose account no is equals to 106.
9. Give name of borrowers having amount greater than 5000.
10. Give name of customers who opened account after date '1-12-96'.
11. Display name of customers whose account no is less than 105.
12. Display name of customer who belongs to either 'NAGPUR' Or 'DELHI'. (**OR & IN**)
13. Display name of customers with branch whose amount is greater than 4000 and account no is less than 105.
14. Find all borrowers whose amount is greater than equals to 3000 & less than equals to 8000. (**AND & BETWEEN**)
15. Find all depositors who do not belongs to 'ANDHERI' branch.
16. Display Account No, Customer Name & Amount of such customers who belongs to 'AJNI', 'KAROLBAGH' Or 'M.G.ROAD' and Account No is less than 104.
17. Display all the details of first five customers.
18. Display all the details of first three depositors whose amount is greater than 1000.

19. Display Loan No, Customer Name of first five borrowers whose branch name does not belongs to 'ANDHERI'.
20. Retrieve all unique cities using DISTINCT. **(**Use **Customers Table)**
21. Retrieve all unique branches using DISTINCT. **(**Use **Branch Table)**
22. Retrieve all the records of customer table as per their city name in ascending order.
23. Retrieve all the records of deposit table as per their amount column in descending order.
24. Update deposit amount of all customers from 3000 to 5000.
25. Change branch name of ANIL from VRCE to C.G. ROAD. (Use **Borrow Table**)
26. Update Account No of SANDIP to 111 & Amount to 5000.
27. Give 10% Increment in Loan Amount.
28. Update deposit amount of all depositors to 5000 whose account no between 103 & 107.
29. Update amount of loan no 321 to *NULL*.
30. Display the name of borrowers whose amount is *NULL*.

Part B:

Create table as per following.

STUDENT

RollNo	Name	Birthdate	SPI	City	Backlog	Branch
101	Keyur	5-1-92	8.5	Rajkot	2	CE
102	Hardik	15-2-94	9.0	Ahmedabad	0	CE
103	Kajal	14-3-96	10.00	Baroda	0	IT
104	Bhoomi	23-6-95	8.90	Ahmedabad	1	ICT
105	Harmit	15-2-94	9.80	Rajkot	1	IT
106	Jay	15-2-94	7.9	Rajkot	2	CE

From the above given tables perform the following SQL queries (SELECT Operation):

1. Give RollNo and Name of students, whose RollNo is greater than 103 and backlog is greater than 0 and branch is either CE or IT.
2. Give name of students whose SPI is between 8 and 9 and branch is either CE or IT. (**OR & IN**)
3. Find all students who do not belongs to 'CE' branch.
4. Display RollNo and Name of first three students.
5. Display all the details of first three students whose SPI is greater than 8.5.
6. Retrieve all unique cities using DISTINCT.
7. Retrieve all unique branches using DISTINCT.
8. Retrieve all the records of student table as per their Backlog in descending order and then SPI in ascending order.
9. Update the branch and city of Jay to MCA and Jamangar respectively.
10. Update the backlog of Keyur and Bhoomi to *NULL*.
11. Display the name of students whose backlog is *NULL* and backlog is greater than 1 and branch is either CE or IT.

2

Part A:

Create table as per following.

EMPLOYEE

EmpNo	EmpName	JoiningDate	Salary	City
101	Keyur	5-1-02	12000.00	Rajkot
102	Hardik	15-2-04	14000.00	Ahmedabad
103	Kajal	14-3-06	15000.00	Baroda
104	Bhoomi	23-6-05	12500.00	Ahmedabad
102	Harmit	15-2-04	14000.00	Rajkot

From the above given tables perform the following SQL queries (DELETE Operation):

1. Display the name of employee whose salary is greater than 13000 and city is either Rajkot or Baroda.
2. Display the name of employee in ascending order by their name.
3. Retrieve all unique cities.
4. Update the city of Keyur and Bhoomi to *NULL*.
5. Display the name of employee whose city is *NULL*.
6. Delete all the records of Employee table having salary greater than and equals to 14000.
7. Delete all the Employees who belongs to 'RAJKOT' city.
8. Delete all the Employees who joined after 1-1-2007.
9. Delete all the records of Employee table. (Use **Truncate**)
10. Remove Employee table. (Use **Drop**)
11. Delete all the records of DEPOSIT table. (Use **Truncate**)
12. Remove DEPOSIT table. (Use **Drop**)
13. Remove BRANCH table. (Use **Drop**)
14. Remove CUSTOMERS table. (Use **Drop**)
15. Remove BORROW table. (Use **Drop**)

Part B:

Create table as per following.

ACCOUNT

ACTNO	CNAME	BNAME	AMOUNT	ADATE
101	ANIL	VRCE	1000.00	1-3-95
102	SUNIL	AJNI	5000.00	4-1-96
103	MEHUL	KAROLBAGH	3500.00	17-11-95
104	MADHURI	CHANDI	1200.00	17-12-95
105	PRMOD	M.G. ROAD	3000.00	27-3-96
106	SANDIP	ANDHERI	2000.00	31-3-96
107	SHIVANI	VIRAR	1000.00	5-9-95
108	KRANTI	NEHRU PLACE	5000.00	2-7-95
109	MINU	POWAI	7000.00	10-8-95

From the above given tables perform the following SQL queries:

1. Retrieve all unique BNAME.
2. Display the Cname in ascending order by their amount and if amount is same then in descending order by cname.
3. Update the BNAME of Anil and Shivani to *NULL*.
4. Display the Cname of customers whose Bname is *NULL*.
5. Delete all the records of Account table having amount greater than and equals to 4000.

6. Delete all the accounts Bname is CHANDI.
7. Delete all the accounts having adate after 1-10-1995.
8. Delete all the records of Account table. (Use **Truncate**)
9. Remove Account table. (Use **Drop**)

Part C:

Create table as per following.

ACCOUNT

ACTNO	CNAME	BNAME	AMOUNT	ADATE
101	ANIL	VRCE	1000.00	1-3-95
102	SUNIL	AJNI	5000.00	4-1-96
103	MEHUL	KAROLBAGH	3500.00	17-11-95
104	MADHURI	CHANDI	1200.00	17-12-95
105	PRMOD	M.G. ROAD	3000.00	27-3-96
106	SANDIP	ANDHERI	2000.00	31-3-96
107	SHIVANI	VIRAR	1000.00	5-9-95
108	KRANTI	NEHRU PLACE	5000.00	2-7-95
109	MINU	POWAI	7000.00	10-8-95

From the above given tables perform the following SQL queries:

1. Display the Cname whose Bname is either AJNI or CHANDI and amount is greater than 3000 and sort the result in ascending order by their amount and if amount is same then in descending order by cname.
2. Retrieve top 3 unique BNAME and sort them in ascending order on BNAME.
3. Display the Cname whose ACTNO is greater than 103 and sort the result in ascending order by their amount and if amount is same then in descending order by cname.
4. Update the BNAME of Anil, Mehul and Shivani to *NULL*.
5. Display the Cname of customers whose Bname is *NULL*.
6. Update the amount of Anil to 5000.
7. Update amount of actno 109 to *NULL*.
8. Retrieve all the records of account table as per their bname in descending order.
9. Delete all the records of Account table. (Use **Truncate**)
10. Remove Account table. (Use **Drop**)

3

Part A:

Create table as per following.

STUDENT

StuID	FirstName	LastName	Website	City	Division
1011	Keyur	Patel	techonthenet.com	Rajkot	II-BCX
1022	Hardik	Shah	digminecraft.com	Ahmedabad	I-BCY
1033	Kajal	Trivedi	bigactivities.com	Baroda	IV-DCX
1044	Bhoomi	Gajera	checkyourmath.com	Ahmedabad	III-DCW
1055	Harmit	Mitel	NULL	Rajkot	II-BCY
1066	Ashok	Jani	NULL	Baroda	II-BCZ

From the above given tables perform the following SQL queries (LIKE Operation):

1. Display the name of students whose name starts with 'k'.
2. Display the name of students whose name consists of five characters.
3. Retrieve the first name & last name of students whose city name ends with a & contains six characters.
4. Display all the students whose last name ends with 'tel'.

5. Display all the students whose first name starts with 'ha' & ends with't'.
6. Display all the students whose first name starts with 'k' and third character is 'y'.
7. Display the name of students having no website and name consists of five characters.
8. Display all the students whose last name consist of 'jer'.
9. Display all the students whose city name starts with either 'r' or 'b'.
10. Display all the name students having websites.
11. Display all the students whose name starts from alphabet A to H.
12. Display all the students whose name's second character is vowel.
13. Display student's name whose city name consist of 'rod'.
14. Retrieve the First & Last Name of students whose website name starts with 'bi'.
15. Display student's city whose last name consists of six characters.
16. Display all the students whose city name consist of five characters & not starts with 'ba'.
17. Show all the student's whose division starts with 'II'.
18. Find out student's first name whose division contains 'bc' anywhere in division name.
19. Show student id and city name in which division consist of six characters and having website name.
20. Display all the students whose name's third character is consonant.

Part B:

Create table as per following.

CUSTOMER

CID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitucion 2222	Mexico D.F.	05021	Mexico
3	Antonio Moreno Taqueria	Antonio Moreno	Mataderos 2312	Mexico D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbkop	Christina Berglund	Berguvsvagen 8	Lulea	S-958 22	Sweden

From the above given tables perform the following SQL queries (LIKE Operation):

1. Return all customers from a city that starts with 'L' followed by one wildcard character, then 'nd' and then two wildcard characters.
2. Return all customers from a city that contains the letter 'L'.
3. Return all customers from a city that do not contains the letter 'L'.
4. Return all customers that starts with 'La'.
5. Return all customers that do not starts with 'La'.
6. Return all customers that starts with 'a' or starts with 'b'.
7. Return all customers that starts with 'a' or starts with 'c' or starts with 't'.
8. Return all customers that starts with 'a' to 'd'.
9. Return all customers that ends with 'a'.
10. Return all customers that do not ends with 'a'.
11. Return all customers that starts with 'b' and ends with 's'.
12. Return all customers that contains the phrase 'or'.
13. Return all customers that starts with "a" and are at least 3 characters in length.

14. Return all customers that have "r" in the second position.
15. Return all customers from Spain.

Part C:

Create table as per following.

CUSTOMER

CID	Name	Age	Address	Salary
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	Hyderabad	4500.00
7	Muffy	24	Indore	10000.00

From the above given tables perform the following SQL queries (LIKE Operation):

1. Display all the records from the CUSTOMERS table, where the SALARY starts with 200.
2. Displays all the records from the CUSTOMERS table with the NAME that has 'al' in any position.
3. Display all the records from the CUSTOMERS table where the Name starts with K and is at least 4 characters in length.
4. Display all the records from the CUSTOMERS table, where the NAME has 'm' in the third position.
5. Retrieves the records of the customers whose name starts with C and ends with i, or customers whose name ends with k.
6. Retrieves all the customers whose name does not start with K.

4

Part A:

Create table as per following.

EMPLOYEE

EID	EName	Department	Salary	JoiningDate	City
101	Rahul	Admin	56000	1-Jan-90	Rajkot
102	Hardik	IT	18000	25-Sep-90	Ahmedabad
103	Bhavin	HR	25000	14-May-91	Baroda
104	Bhoomi	Admin	39000	8-Feb-91	Rajkot
105	Rohit	IT	17000	23-Jul-90	Jamnagar
106	Priya	IT	9000	18-Oct-90	Ahmedabad
107	Neha	HR	34000	25-Dec-91	Rajkot

From the above given tables perform the following SQL queries:

1. Display the Highest, Lowest, Total, and Average salary of all employees. Label the columns Maximum, Minimum, Total_Sal and Average_Sal, respectively.
2. Find total number of employees of EMPLOYEE table.
3. Give maximum salary from IT department.
4. Count total number of cities of employee without duplication.
5. Display city with the total number of employees belonging to each city.
6. Display city having more than one employee.
7. Give total salary of each department of EMPLOYEE table.
8. Give average salary of each department of EMPLOYEE table without displaying the respective department name.

9. Give minimum salary of employee who belongs to Ahmedabad.
10. List the departments having total salaries more than 50000 and located in city Rajkot.
11. Count the number of employees living in Rajkot.
12. Display the difference between the highest and lowest salaries. Label the column DIFFERENCE.
13. Display the total number of employees hired before 1st January, 1991.
14. Display total salary of each department with total salary exceeding 35000 and sort the list by total salary.
15. List out department names in which more than two employees.

Part B:

Create table as per following.

COMPANY

Title	Company	Type	Production_year	System	Production_cost	Revenue	Rating
Blasting Boxes	Simone Games	action adventure	1998	PC	100000	200000	7
Run Run Run!	13 Mad Bits	shooter	2011	PS3	3500000	650000	3
Duck n'Go	13 Mad Bits	shooter	2012	Xbox	3000000	1500000	5
SQL Wars!	Vertabelo	wargames	2017	Xbox	5000000	25000000	10
Tap Tap Hex!	PixelGaming Inc.	rhythm	2006	PS2	2500000	3500000	7
NoRisk	Simone Games	action adventure	2004	PS2	1400000	3400000	8

From the above given tables perform the following SQL queries:

1. Display the name and total revenue for each company.
2. Generate a report with the production year and the number of games released this year (named count), the average of production cost for all games produced in this year (named avg_cost) and the average revenue for that year (named avg_revenue).
3. Count how many games of a given type are profitable (i.e. the revenue was greater than the production cost). Show the game type and the number of profitable games (named number_of_games) for each type.
4. Obtain the type of games and the total revenue generated for games with a production_year after 2010 and with a PS2 or PS3 system. Order the result so the types with the highest revenue come first.
5. For all companies present in the table, obtain their names and the sum of gross profit over all years. (Assume that gross profit = revenue - cost of production). Name this column gross_profit_sum. Order the results by gross profit, in descending order.
6. Obtain the yearly gross profit of each company. In other words, we want a report with the company name, the year, and the gross profit for that year. Order the report by company name and year.
7. For each company, select its name, the number of games it's produced (as the number_of_games column), and the average cost of production (as the avg_cost column). Show only companies producing more than one game.

5

Part A:

Create table as per following.

ORDERS

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	05-10-2012	3005	5002
70009	270.65	10-09-2012	3001	5005
70002	65.26	05-10-2012	3002	5001
70004	110.5	17-08-2012	3009	5003
70007	948.5	10-09-2012	3005	5002
70005	2400.6	27-07-2012	3007	5001
70008	5760	10-09-2012	3002	5001
70010	1983.43	10-10-2012	3004	5006
70003	2480.4	10-10-2012	3009	5003
70012	250.45	27-06-2012	3008	5002
70011	75.29	17-08-2012	3003	5007
70013	3045.6	25-04-2012	3002	5001
70001	150.5	05-10-2012	3005	5002
70009	270.65	10-09-2012	3001	5005
70002	65.26	05-10-2012	3002	5001

CUSTOMER

customer_id	cust_name	city	Grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

SALESMAN

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

From the above given tables perform the following SQL queries:

1. Write a SQL query to calculate total purchase amount of all orders. Return total purchase amount.
2. Write a SQL query to calculate the average purchase amount of all orders. Return average purchase amount.
3. Write a SQL query that counts the number of unique salespeople. Return number of salespeople.
4. Write a SQL query to count the number of customers. Return number of customers.
5. Write a SQL query to determine the number of customers who received at least one grade for their activity.
6. Write a SQL query to find the maximum purchase amount.
7. Write a SQL query to find the minimum purchase amount.

8. Write a SQL query to find the highest grade of the customers in each city. Return city, maximum grade.
9. Write a SQL query to find the highest purchase amount ordered by each customer. Return customer ID, maximum purchase amount.
10. Write a SQL query to find the highest purchase amount ordered by each customer on a particular date. Return, order date and highest purchase amount.

Part B:

1. Write a SQL query to determine the highest purchase amount made by each salesperson on '2012-08-17'. Return salesperson ID, purchase amount.
2. Write a SQL query to find the highest order (purchase) amount by each customer on a particular order date. Filter the result by highest order (purchase) amount above 2000.00. Return customer id, order date and maximum purchase amount.
3. Write a SQL query to find the maximum order (purchase) amount in the range 2000 - 6000 (Begin and end values are included.) by combination of each customer and order date. Return customer id, order date and maximum purchase amount.
4. Write a SQL query to find the maximum order (purchase) amount based on the combination of each customer and order date. Filter the rows for maximum order (purchase) amount is either 2000, 3000, 5760, 6000. Return customer id, order date and maximum purchase amount.
5. Write a SQL query to determine the maximum order amount for each customer. The customer ID should be in the range 3002 and 3007(Begin and end values are included.). Return customer id and maximum purchase amount.
6. Write a SQL query to find the maximum order (purchase) amount for each customer. The customer ID should be in the range 3002 and 3007(Begin and end values are included.). Filter the rows for maximum order (purchase) amount is higher than 1000. Return customer id and maximum purchase amount.
7. Write a SQL query to determine the maximum order (purchase) amount generated by each salesperson. Filter the rows for the salesperson ID is in the range 5003 and 5008 (Begin and end values are included.). Return salesperson id and maximum purchase amount.
8. Write a SQL query to count all the orders generated on '2012-08-17'. Return number of orders.
9. Write a SQL query to count the number of salespeople in a city. Return number of salespeople.
10. Write a SQL query to count the number of orders based on the combination of each order date and salesperson. Return order date, salesperson id.

Part C:

1. Write a SQL query to calculate the average product price. Return average product price.
2. Write a SQL query to count the number of products whose price are higher than or equal to 350. Return number of products.
3. Write a SQL query to compute the average price for unique companies. Return average price and company id.
4. Write a SQL query to compute the sum of the allotment amount of all departments. Return sum of the allotment amount.
5. Write a SQL query to count the number of employees in each department. Return department code and number of employees.

6

Part-A:

Create table as per following.

STUDENT

Rno	Name	Branch
101	Raju	CE
102	Amit	CE
103	Sanjay	ME
104	Neha	EC
105	Meera	EE
106	Mahesh	ME

RESULT

Rno	SPI
101	8.8
102	9.2
103	7.6
104	8.2
105	7.0
107	8.9

EMPLOYEE

EmployeeNo	Name	ManagerNo
E01	Tarun	*NULL*
E02	Rohan	E02
E03	Priya	E01
E04	Milan	E03
E05	Jay	E01
E06	Anjana	E04

From the above given tables perform the following queries (Join):

1. Combine information from student and result table using cross join or Cartesian product.
2. Display Rno, Name, Branch and SPI of all students.
3. Display Rno, Name, Branch and SPI of CE branch's student only.
4. Display Rno, Name, Branch and SPI of other than EC branch's student only.
5. Display average result of each branch.
6. Display average result of each branch and sort them in ascending order by SPI.
7. Display average result of CE and ME branch.
8. Perform the left outer join on Student and Result tables.
9. Perform the right outer join on Student and Result tables.
10. Perform the full outer join on Student and Result tables.
11. Retrieve the names of employee along with their manager name from the Employee table.

Part-B:

Create table as per following.

PERSON

PersonID	PersonName	DepartmentID	Salary	JoiningDate	City
101	Rahul Tripathi	2	56000	01-01-2000	Rajkot
102	Hardik Pandya	3	18000	25-09-2001	Ahmedabad
103	Bhavin Kanani	4	25000	14-05-2000	Baroda
104	Bhoomi Vaishnav	1	39000	08-02-2005	Rajkot
105	Rohit Topiya	2	17000	23-07-2001	Jamnagar
106	Priya Menpara	NULL	9000	18-10-2000	Ahmedabad
107	Neha Sharma	2	34000	25-12-2002	Rajkot
108	Nayan Goswami	3	25000	01-07-2001	Rajkot
109	Mehul Bhundiya	4	13500	09-01-2005	Baroda
110	Mohit Maru	5	14000	25-05-2000	Jamnagar

DEPARTMENT

DepartmentID	DepartmentName	DepartmentCode	Location
1	Admin	Admin	A-Block
2	Computer	CE	C-Block
3	Civil	CI	G-Block
4	Electrical	EE	E-Block
5	Mechanical	ME	B-Block

From the above given table perform the following SQL queries (Join & Group By):

1. Find all persons with their department name & code.
2. Give department wise maximum & minimum salary with department name.
3. Find all departments whose total salary is exceeding 100000.
4. Retrieve person name, salary & department name who belongs to Jamnagar city.
5. Find all persons who does not belongs to any department.
6. Find department wise person counts.
7. Find average salary of person who belongs to Ahmedabad city.
8. Produce Output Like: <PersonName> earns <Salary> from department <DepartmentName> monthly. (In Single Column)
9. List all departments who have no persons.
10. Find city & department wise total, average & maximum salaries.

Part – C:

1. Display Unique city names.
2. List out department names in which more than two persons.
3. Combine person name's first three characters with city name's last three characters in single column.
4. Give 10% increment in Computer department employee's salary.
5. Display all the person name's who's joining dates difference with current date is more than 365 days.

7

Part – A:

Create Database with Name**: Person_Info**

Create following table under Person_Info database**. (Using Design Mode)**

PERSON

Column_Name	DataType	Constraints
PersonID	Int	Primary Key
PersonName	Varchar (100)	Not Null
DepartmentID	Int	Foreign Key, Null
Salary	Decimal (8,2)	Not Null
JoiningDate	Datetime	Not Null
City	Varchar (100)	Not Null

DEPARTMENT

Column_Name	DataType	Constraints
DepartmentID	Int	Primary Key
DepartmentName	Varchar (100)	Not Null, Unique
DepartmentCode	Varchar (50)	Not Null, Unique
Location	Varchar (50)	Not Null

PersonID	PersonName	DepartmentID	Salary	JoiningDate	City
101	Rahul Tripathi	2	56000	01-01-2000	Rajkot
102	Hardik Pandya	3	18000	25-09-2001	Ahmedabad
103	Bhavin Kanani	4	25000	14-05-2000	Baroda
104	Bhoomi Vaishnav	1	39000	08-02-2005	Rajkot
105	Rohit Topiya	2	17000	23-07-2001	Jamnagar
106	Priya Menpara	NULL	9000	18-10-2000	Ahmedabad
107	Neha Sharma	2	34000	25-12-2002	Rajkot
108	Nayan Goswami	3	25000	01-07-2001	Rajkot
109	Mehul Bhundiya	4	13500	09-01-2005	Baroda
110	Mohit Maru	5	14000	25-05-2000	Jamnagar

DepartmentID	DepartmentName	DepartmentCode	Location
1	Admin	Adm	A-Block
2	Computer	CE	C-Block
3	Civil	CI	G-Block
4	Electrical	EE	E-Block
5	Mechanical	ME	B-Block

Part – B:

From the above given table perform the following SQL queries (Join):

1. Find all persons with their department name & code.
2. Find the person's name whose department is located in C-Block.
3. Retrieve person name, salary & department name who belongs to Jamnagar city.
4. Retrieve person name, salary & department name who does not belong to Rajkot city.
5. Retrieve person's name of the person who joined the Civil department after 1-Aug-2001.
6. Find details of all persons who belong to the Computer department.
7. Display all the person's name with the department whose joining date difference with the current date is more than 365 days.
8. Find department wise person counts.
9. Give department wise maximum & minimum salary with department name.
10. Find city wise total, average, maximum and minimum salary.
11. Find the average salary of a person who belongs to Ahmedabad city.
12. Produce Output Like: <PersonName> lives in <City> and works in <DepartmentName> Department. (In single column)
13. Produce Output Like: <PersonName> earns <Salary> from <DepartmentName> department monthly. (In single column)
14. Find city & department wise total, average & maximum salaries.
15. Find all persons who do not belong to any department.

Part – C:

1. Find all departments whose total salary is exceeding 100000.
2. List all departments who have no person.
3. List out department names in which more than two persons are working.
4. Give a 10% increment in the Computer department employee's salary. (Use Update)
5. Calculate Employee Experience in Years, Months & Days with respect to their joining Date.

8 **Part A:**

Create table as per following.

ORDERS

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	05-10-2012	3005	5002
70009	270.65	10-09-2012	3001	5005
70002	65.26	05-10-2012	3002	5001
70004	110.5	17-08-2012	3009	5003
70007	948.5	10-09-2012	3005	5002
70005	2400.6	27-07-2012	3007	5001
70008	5760	10-09-2012	3002	5001
70010	1983.43	10-10-2012	3004	5006

70003	2480.4	10-10-2012	3009	5003
70012	250.45	27-06-2012	3008	5002
70011	75.29	17-08-2012	3003	5007
70013	3045.6	25-04-2012	3002	5001
70001	150.5	05-10-2012	3005	5002
70009	270.65	10-09-2012	3001	5005
70002	65.26	05-10-2012	3002	5001

SALESMAN

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

CUSTOMER

customer_id	cust_name	city	Grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

From the above given tables perform the following SQL queries (Join):

1. Write a SQL query to find the salesperson and customer who reside in the same city. Return Salesman, cust_name and city.
2. Write a SQL query to find those orders where the order amount exists between 500 and 2000. Return ord_no, purch_amt, cust_name, city.
3. Write a SQL query to find the salesperson(s) and the customer(s) he represents. Return Customer Name, city, Salesman, commission.
4. Write a SQL query to find salespeople who received commissions of more than 12 percent from the company. Return Customer Name, customer city, Salesman, commission.
5. Write a SQL query to locate those salespeople who do not live in the same city where their customers live and have received a commission of more than 12% from the company. Return Customer Name, customer city, Salesman, salesman city, commission.
6. Write a SQL query to find the details of an order. Return ord_no, ord_date, purch_amt, Customer Name, grade, Salesman, commission.
7. Write a SQL statement to join the tables salesman, customer and orders so that the same column of each table appears once and only the relational rows are returned.
8. Write a SQL query to display the customer name, customer city, grade, salesman, salesman city. The results should be sorted by ascending customer_id.
9. Write a SQL query to find those customers with a grade less than 300. Return cust_name, customer city, grade, Salesman, salesmancity. The result should be ordered by ascending customer_id.

10. Write a SQL statement to make a report with customer name, city, order number, order date, and order amount in ascending order according to the order date to determine whether any of the existing customers have placed an order or not.

Part-B:

1. Write a SQL statement to generate a report with customer name, city, order number, order date, order amount, salesperson name, and commission to determine if any of the existing customers have not placed orders or if they have placed orders through their salesman or by themselves.
2. Write a SQL statement to generate a list in ascending order of salespersons who work either for one or more customers or have not yet joined any of the customers.
3. Write a SQL query to list all salespersons along with customer name, city, grade, order number, date, and amount.
4. Write a SQL statement to make a list for the salesmen who either work for one or more customers or yet to join any of the customer. The customer may have placed, either one or more orders on or above order amount 2000 and must have a grade, or he may not have placed any order to the associated supplier.
5. For those customers from the existing list who put one or more orders, or which orders have been placed by the customer who is not on the list, create a report containing the customer name, city, order number, order date, and purchase amount.

Part-C:

1. Write a SQL statement to generate a report with the customer name, city, order no. order date, purchase amount for only those customers on the list who must have a grade and placed one or more orders or which order(s) have been placed by the customer who neither is on the list nor has a grade.
2. Write a SQL query to combine each row of the salesman table with each row of the customer table.
3. Write a SQL statement to create a Cartesian product between salesperson and customer, i.e. each salesperson will appear for all customers and vice versa for that salesperson who belongs to that city.
4. Write a SQL statement to create a Cartesian product between salesperson and customer, i.e. each salesperson will appear for every customer and vice versa for those salesmen who belong to a city and customers who require a grade.
5. Write a SQL statement to make a Cartesian product between salesman and customer i.e. each salesman will appear for all customers and vice versa for those salesmen who must belong to a city which is not the same as his customer and the customers should have their own grade.

9 **Part – A: Create table as per following.**

CITY

CityID (Primary Key)	Name (Unique Key)	Pincode (Not Null)	Remakrs
1	Rajkot	360005	Good
2	Surat	335009	Very Good
3	Baroda	390001	Awesome
4	Jamnagar	361003	Smart
5	Junagadh	362229	Historic
6	Morvi	363641	Ceramic

VILLAGE

VID (Primary Key)	Name (Not Null)	CityID (Foreign Key)
101	Raiya	1
102	Madhapar	1
103	Dodka	3
104	Falla	4
105	Bhesan	5
106	Dhoraji	5

From the above given tables perform the following SQL queries:

1. Display all the villages of Rajkot city.
2. Display city along with their villages & pin code.
3. Display the city having more than one village.
4. Display the city having no village.
5. Count the total number of villages in each city.

6. Count the number of cities having more than one village.

Create below table Student with following constraints

1. Do not allow SPI more than 10 and less than 0.
2. Do not allow Bklog less than 0.
3. Enter the default value as 'General' in branch to all new records IF no other value is specified.

STUDENT

Rno(PK)	Name	Branch	SPI	Bklog
101	Raju	CE	8.80	0
102	Amit	CE	2.20	3
103	Sanjay	ME	1.50	6
104	Neha	EC	7.65	0
105	Meera	EE	5.52	2
106	Mahesh	General	4.50	3

- ► Try to update SPI of Raju from 8.80 to 12.
- ► Try to update Bklog of Neha from 0 to -1.

Part - B: Create table as per following schema with proper validation and try to insert data which violate your validation.

1. Emp(Eid, Ename, Did, Cid, Salary, Experience)
   Dept(Did, Dname)
   City(Cid, Cname)

Part - C: Create table as per following schema with proper validation and try to insert data which violate your validation.

1. Emp(Eid, Ename, Did, Cid, Salary, Experience)
   Dept(Did, Dname)
   City(Cid, Cname, Did))
   District(Did, Dname, Sid)
   State(Sid, Sname, Cid)
   Country(Cid, Cname)
2. Insert 5 records in each table.
3. Display employeename, departmentname, Salary, Experience, City, District, State and country of all employees.

10

Part - A:

Create table as per following.

STUDENT

Rno	Name	City	DID
101	Raju	Rajkot	10
102	Amit	Ahmedabad	20
103	Sanjay	Baroda	40
104	Neha	Rajkot	20
105	Meera	Ahmedabad	30
106	Mahesh	Baroda	10

ACADEMIC

Rno	SPI	Bklog
101	8.8	0
102	9.2	2
103	7.6	1
104	8.2	4
105	7.0	2
106	8.9	3

DEPARTMENT

DID	DName
10	Computer
20	Electrical
30	Mechanical
40	Civil

From the above given tables perform the following SQL queries (Sub Queries):

1. Display details of students who are from computer department.

2. Displays name of students whose SPI is more than 8.
3. Display details of students of computer department who belongs to Rajkot city.
4. Find total number of students of electrical department.
5. Display name of student who is having maximum SPI.
6. Display details of students having more than 1 backlog.
7. Display name of student who is having second highest SPI.
8. Display name of students who are either from computer department or from mechanical department.
9. Display name of students who are in same department as 102 studying in.
10. Display name of students whose SPI is more than 9 and who is from electrical department.

Part - B: **Create table as per following.**

COMPANY_MASTER

COM_ID	COM_NAME
11	Samsung
12	iBall
13	Epsion
14	Zebronics
15	Asus
16	Frontech

ITEM_MASTER

PRO_ID	PRO_NAME	PRO_PRICE	PRO_COM
101	Mother Board	3200	15
102	Key Board	450	16
103	ZIP drive	250	14
104	Speaker	550	16
105	Monitor	5000	11
106	DVD drive	900	12
107	CD drive	800	12
108	Printer	2600	13
109	Refill cartridge	350	13
110	Mouse	250	12

EMP_DETAILS

EMP_IDNO	EMP_FNAME	EMP_LNAME	EMP_DEPT
127323	Michale	Robbin	57
526689	Carlos	Snares	63
843795	Enric	Dosio	57
328717	Jhon	Snares	63
444527	Joseph	Dosni	47
659831	Zanifer	Emily	47
847674	Kuleswar	Sitaraman	57
748681	Henrey	Gabriel	47
555935	Alex	Manuel	57
539569	George	Mardy	27
733843	Mario	Saule	63
631548	Alan	Snappy	27
839139	Maria	Foster	57

EMP_DEPARTMENT

DPT_CODE	DPT_NAME	DPT_ALLOTMENT
57	IT	65000
63	Finance	15000
47	HR	240000
27	RD	55000
89	QC	75000

From the above given tables perform the following SQL queries:

1. Write a SQL query to calculate the average price of each manufacturer's product along with their name. Return Average Price and Company.
2. Write a SQL query to calculate the average price of each manufacturer's product of 350 or more. Return Average Price and Company.
3. Write a SQL query to find the most expensive product of each company. Return Product Name, Price and Company.
4. Write a SQL query to find employees whose last name is Gabriel or Dosio. Return emp_idno, emp_fname, emp_lname and emp_dept.
5. Write a SQL query to find the employees who work in department 89 or 63. Return emp_idno, emp_fname, emp_lname and emp_dept.

Part - C:

1. Write a SQL query to find those employees who work for the department where the departmental allotment amount is more than Rs. 50000. Return emp_fname and emp_lname.
2. Write a SQL query to find the departments whose sanction amount is higher than the average sanction amount for all departments. Return dpt_code, dpt_name and dpt_allotment.
3. Write a SQL query to find which departments have more than two employees. Return dpt_name.
4. Write a SQL query to find the departments with the second lowest sanction amount. Return emp_fname and emp_lname

11 **Part - A**: **Create table as per following**.

ORDERS

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	05-10-2012	3005	5002
70009	270.65	10-09-2012	3001	5005
70002	65.26	05-10-2012	3002	5001
70004	110.5	17-08-2012	3009	5003
70007	948.5	10-09-2012	3005	5002
70005	2400.6	27-07-2012	3007	5001
70008	5760	10-09-2012	3002	5001
70010	1983.43	10-10-2012	3004	5006
70003	2480.4	10-10-2012	3009	5003
70012	250.45	27-06-2012	3008	5002
70011	75.29	17-08-2012	3003	5007
70013	3045.6	25-04-2012	3002	5001
70001	150.5	05-10-2012	3005	5002
70009	270.65	10-09-2012	3001	5005
70002	65.26	05-10-2012	3002	5001

SALESMAN

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

CUSTOMER

customer_id	cust_name	city	Grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London		5005

From the above given tables perform the following queries:

1. Write a SQL query to find all the orders issued by the salesman 'Paul Adam'. Return ord_no, purch_amt, ord_date, customer_id and salesman_id.
2. Write a SQL query to find all orders generated by London-based salespeople. Return ord_no, purch_amt, ord_date, customer_id, salesman_id.
3. Write a SQL query to find all orders generated by the salespeople who may work for customers whose id is 3007. Return ord_no, purch_amt, ord_date, customer_id, salesman_id.
4. Write a SQL query to find the order values greater than the average order value of 10th October 2012. Return ord_no, purch_amt, ord_date, customer_id, salesman_id.
5. Write a SQL query to find all the orders generated in New York city. Return ord_no, purch_amt, ord_date, customer_id and salesman_id.
6. Write a SQL query to determine the commission of the salespeople in Paris. Return commission.
7. Write a query to display all the customers whose ID is 2001 below the salesperson ID of Mc Lyon.
8. write a SQL query to count the number of customers with grades above the average in New York City. Return grade and count.
9. Write a SQL query to find those salespeople who earned the maximum commission. Return ord_no, purch_amt, ord_date, and salesman_id.
10. Write SQL query to find the customers who placed orders on 17th August 2012. Return ord_no, purch_amt, ord_date, customer_id, salesman_id and cust_name.
11. Write a SQL query to find salespeople who had more than one customer. Return salesman_id and name.
12. Write a SQL query to find those orders, which are higher than the average amount of the orders. Return ord_no, purch_amt, ord_date, customer_id and salesman_id.
13. Write a SQL query to find those orders that are equal or higher than the average amount of the orders. Return ord_no, purch_amt, ord_date, customer_id and salesman_id.
14. Write a query to find the sums of the amounts from the orders table, grouped by date, and eliminate all dates where the sum was not at least 1000.00 above the maximum order amount for that date.
15. Write a query to extract all data from the customer table if and only if one or more of the customers in the customer table are located in London. Sample table : Customer

Part - B:

1. Write a SQL query to find salespeople who deal with multiple customers. Return salesman_id, name, city and commission.

2. Write a SQL query to find salespeople who deal with a single customer. Return salesman_id, name, city and commission.
3. Write a SQL query to find the salespeople who deal the customers with more than one order. Return salesman_id, name, city and commission.
4. Write a SQL query to find the salespeople who deal with those customers who live in the same city. Return salesman_id, name, city and commission.
5. Write a SQL query to find salespeople whose place of residence matches any city where customers live. Return salesman_id, name, city and commission.
6. Write a SQL query to find all those salespeople whose names appear alphabetically lower than the customer's name. Return salesman_id, name, city, commission.
7. Write a SQL query to find all those customers with a higher grade than all the customers alphabetically below the city of New York. Return customer_id, cust_name, city, grade, salesman_id.
8. Write a SQL query to find all those orders whose order amount exceeds at least one of the orders placed on September 10th 2012. Return ord_no, purch_amt, ord_date, customer_id and salesman_id.
9. Write a SQL query to find orders where the order amount is less than the order amount of a customer residing in London City. Return ord_no, purch_amt, ord_date, customer_id and salesman_id.
10. Write a SQL query to find those orders where every order amount is less than the maximum order amount of a customer who lives in London City. Return ord_no, purch_amt, ord_date, customer_id and salesman_id.

Part - C:

1. Write a SQL query to find those customers whose grades are higher than those living in New York City. Return customer_id, cust_name, city, grade and salesman_id.
2. Write a SQL query to calculate the total order amount generated by a salesperson. Salespersons should be from the cities where the customers reside. Return salesperson name, city and total order amount.
3. Write a SQL query to find those customers whose grades are not the same as those who live in London City. Return customer_id, cust_name, city, grade and salesman_id.
4. Write a SQL query to find those customers whose grades are different from those living in Paris. Return customer_id, cust_name, city, grade and salesman_id.
5. Write a SQL query to find all those customers who have different grades than any customer who lives in Dallas City. Return customer_id, cust_name,city, grade and salesman_id.

12 **Create table as per following.**

PERSON

Column_Name	DataType	Constraints
WorkerID	Int	Primary Key, Auto Increment
FirstName	Varchar (100)	Not Null
LastName	Varchar (100)	Not Null
Salary	Decimal (8,2)	Not Null
JoiningDate	Datetime	Not Null
DepartmentID	Int	Foreign Key, Null
DesignationID	Int	Foreign Key, Null

Department

Column_Name	DataType	Constraints
DepartmentID	Int	Primary Key
DepartmentName	Varchar (100)	Not Null, Unique

Designation

Column_Name	DataType	Constraints
DesignationID	Int	Primary Key
DesignationName	Varchar (100)	Not Null, Unique

WorkerID	FirstName	LastName	Salary	JoiningDate	DepartmentID	DesignationID
101	Rahul	Anshu	56000	01-01-1990	1	12
102	Hardik	Hinsu	18000	25-09-1990	2	11
103	Bhavin	Kamani	25000	14-05-1991	NULL	11
104	Bhoomi	Patel	39000	20-02-2014	1	13
105	Rohit	Rajgor	17000	23-07-1990	2	15
106	Priya	Mehta	25000	18-10-1990	2	NULL
107	Neha	Trivedi	18000	20-02-2014	3	15

DepartmentID	DepartmentName
1	Admin
2	IT
3	HR
4	Account

DesignationID	DesignationName
11	Jobber
12	Welder
13	Clerk
14	Manager
15	CEO

From the above given tables create Stored Procedures:

Part – A:

1. Create a Procedure on Department, Designation & Person Table for INSERT, UPDATE & DELETE Procedures.
2. Create a Procedure on Department, Designation & Person Table for SELECTBYPRIMARYKEY
3. Create a Procedure on Department, Designation & Person Table (If foreign key is available then do write join and take columns on select list)
4. Create a Procedure that shows details of the first 3 persons.

Part – B:

1. Create a Procedure that takes the department name as input and returns a table with all workers working in that department.
2. Create Procedure that takes department name & designation name as input and returns a table with worker's first name, salary, joining date & department name.
3. Create a Procedure that takes the first name as an input parameter and display all the details of the worker with their department & designation name.
4. Create Procedure which displays department wise maximum, minimum & total salaries.
5. Create Procedure which displays designation wise average & total salaries.

Part – C:

1. Create Procedure that Accepts Department Name and Returns Person Count.
2. Create Procedure that accepts Department Name & Designation as a parameter with given test cases and returns a table with FirstName, LastName, Salary, JoiningDate, DepartmentName & Designation.

Department Name	**Designation**
IT	NULL
NULL	Jobber
IT	Jobber
NULL	NULL

3. Create Procedure that returns DepartmentID, DepartmentName & Count of all person belongs to that department. i.e. 1 | Admin | 2

13 **Create table as per following.**

Person		
Column_Name	**DataType**	**Constraints**
PersonID	Int	Primary Key
PersonName	Varchar (100)	Not Null
Salary	Decimal (8,2)	Not Null

JoiningDate	Datetime	Not Null
City	Varchar (100)	Not Null
Age	Int	Null
BirthDate	Datetime	Not Null

PersonLog		
Column_Name	**DataType**	**Constraints**
PLogID	Int	Primary Key, Auto increment
PersonID	Int	Not Null
PersonName	Varchar (250)	Not Null
Operation	Varchar (50)	Not Null
UpdateDate	Datetime	Not Null

From the above given tables create Triggers:

Part – A:

1. Create a trigger that fires on INSERT, UPDATE and DELETE operation on the Person table to display a message "Record is Affected."
2. Create a trigger that fires on INSERT, UPDATE and DELETE operation on the Person table. For that, log all operations performed on the person table into PersonLog.

Part – B:

1. Create an INSTEAD OF trigger that fires on INSERT, UPDATE and DELETE operation on the Person table. For that, log all operations performed on the person table into PersonLog.
2. Create a trigger that fires on INSERT operation on the Person table to convert person name into uppercase whenever the record is inserted.

Part – C:

1. Create a trigger that fires on INSERT operation on person table, which calculates the age and update that age in Person table.
2. Create DELETE trigger on PersonLog table, when we delete any record of PersonLog table it prints 'Record deleted successfully from PersonLog'.

14 **Create table as per following.**

Products		
Column_Name	**DataType**	**Constraints**
Product_id	Int	Primary Key
Product_Name	Varchar (250)	Not Null
Price	Decimal (10,2)	Not Null

Products		
Product_id	**Product_Name**	**Price**
1	Smatphone	35000
2	Laptop	65000
3	Headphones	5500
4	Television	85000
5	Gaming Console	32000

From the above given tables create Cursors:

Part – A:

1. Create a cursor Product_Cursor to fetch all the rows from a products table.
2. Create a cursor Product_Cursor_Fetch to fetch the records in form of ProductID_ProductName. (Example: 1_Smartphone)
   Create a cursor Product_CursorDelete that deletes all the data from the Products table.

Part – B:

1. Create a cursor Product_CursorUpdate that retrieves all the data from the products table and increases the price by 10%.

Part – C:

1. Create a cursor to insert details of Products into the NewProducts table if the product is "Laptop" (Note: Create NewProducts table first with same fields as Products table)

15

User Defined Functions

Part – A:

1. Write a function to print "hello world".
2. Write a function which returns addition of two numbers.
3. Write a function to print a cube of a given number.
4. Write a function to check whether the given number is ODD or EVEN.
5. Write a function which returns a table with details of a person whose first name starts with B.
6. Write a function which returns a table with unique first names from the person table.

Part – B:

1. Write a function to compare two integers and return the comparison result. (Using Case statement)
2. Write a function to print number from 1 to N. (Using while loop)
3. Write a function to print the sum of even numbers between 1 to 20.

Part – C:

1. Write a function to check whether a given number is prime or not.
2. Write a function which accepts two parameters start date & end date, and returns a difference in days.
3. Write a function which accepts two parameters year & month in integer and returns total days in a given month & year.
4. Write a function which accepts departmentID as a parameter & returns a detail of the persons.

16

Part – A: Create Database with Name: *BANK_INFO* and Create collection as per following.

DEPOSIT

ACTNO	CNAME	BNAME	AMOUNT	ADATE
101	ANIL	VRCE	1000.00	1-3-95
102	SUNIL	AJNI	5000.00	4-1-96
103	MEHUL	KAROLBAGH	3500.00	17-11-95
104	MADHURI	CHANDI	1200.00	17-12-95
105	PRMOD	M.G. ROAD	3000.00	27-3-96
106	SANDIP	ANDHERI	2000.00	31-3-96
107	SHIVANI	VIRAR	1000.00	5-9-95
108	KRANTI	NEHRU PLACE	5000.00	2-7-95

From the above given collection perform the following queries in MongoDB:

1. Retrieve/Display every document of Deposit collection.
2. Retrieve/Display every document of Deposit collection. (Use: pretty())
3. Display only one document of Deposit collection. (Use: findOne())
4. Insert following document to Deposit collection. (Use: insertOne())

109	KIRTI	VIRAR	3000.00	3-5-97

5. Insert following documents to your collection. (Use: insertMany())

110	MITALI	ANDHERI	4500.00	4-9-95
111	RAJIV	NEHRU PLACE	7000.00	2-10-98

6. Display documents with CNAME, BNAME and AMOUNT fields.
7. Display every document of Deposit collection on ascending order by CNAME.
8. Display every document of Deposit collection on descending order by BNAME.
9. Display every document of Deposit collection on ascending order by ACTNO and descending order by AMOUNT.

10. Display only two documents of Deposit collection.
11. Display 3rd document of Deposit collection.
12. Display 6th and 7th documents of Deposit collection.
13. Display the count of documents in Deposit collection.
14. Display only first documents of Deposit collection.
15. Display every document of Deposit collection on descending order by AMOUNT.

Part- B:

1. Insert following document to Deposit collection. (Use: insertOne())

112	MANISH	ANDHERI	8000.00	9-5-98

2. Display 9th document of Deposit collection.
3. Display 11th and 12th documents of Deposit collection.

Part – C:

1. Display every document of Deposit collection on ascending order by AMOUNT and descending order by BNAME.
2. Display only five documents of Deposit collection.
3. Delete all the documents of collection Deposit.
4. Drop BANK_INFO database.

17

Part – A: Create collection as per following.

EMPLOYEE

EID	ENAME	GENDER	JOININGDATE	SALARY	CITY
1	Nick	Male	01-JAN-13	4000	London
2	Julian	Female	01-OCT-14	3000	New York
3	Roy	Male	01-JUN-16	3500	London
4	Tom	Male	NULL	4500	London
5	Jerry	Male	01-FEB-13	2800	Sydney
6	Philip	Male	01-JAN-15	7000	New York
7	Sara	Female	01-AUG-17	4800	Sydney
8	Emily	Female	01-JAN-15	5500	New York
9	Michael	Male	NULL	6500	London
10	John	Male	01-JAN-15	8800	London

From the above given collection perform the following queries in MongoDB:

1. Display employees whose gender is Male.
2. Display employees who belong to London city.
3. Display employees whose salary is greater than 3500.
4. Display employees whose joining date is before 2015-01-01.
5. Display employees whose EID is greater than or equal to 7.
6. Display employees whose city is Landon or New York (use:IN)
7. Display employees who do not belongs to Landon or New York (use: NOT IN)
8. Display the EID of those employee who lives in city London.
9. Display first 2 employee names who lives in New york.
10. Display next 2 employee after skipping first 2 whose city is London.
11. Display Male employees who lives Sydney.
12. Display EID, ENAME, CITY and SALARY of those employees who belongs to London or Sydney.
13. Display ENAME, SALARY, and CITY of those employee whose salary is more than 7000.
14. Display documents whose name start with E.
15. Display documents whose name starts with S or M in your collection.
16. Display documents where city starts with A to M in your collection.
17. Display documents where city name ends in 'ney'.
18. Display employee info whose name contains n. (Both uppercase(N) and lowercase(n))
19. Display employee info whose name starts with E and having 5 characters.

20. Display employee whose name start with S and ends in a.
21. Display EID, ENAME, CITY and SALARY whose name starts with 'Phi'.
22. Display ENAME, JOININGDATE and CITY whose city contains 'dne' as three letters in city name.
23. Display ENAME, JOININGDATE and CITY who does not belongs to city London or Sydney.
24. Delete the documents whose city is New York.
25. Update ENAME of Nick to 'Naysa' and GENDER to 'Female'.

Part – B: Create collection as per following.

STUDENT

ROLLNO	SNAME	DEPARTMENT	FEES	SEM	GENDER	CITY
101	Vina	CE	15000	3	Female	Rajkot
102	Krisha	EC	8000	5	Female	Ahmedabad
103	Priti	Civil	12000	7	Female	Baroda
104	Mitul	CE	15000	3	Male	Rajkot
105	Keshav	CE	15000	3	Male	Jamnagar
106	Zarna	Civil	12000	5	Female	Ahmedabad
107	Nima	EE	9000	5	Female	Rajkot
108	Dhruv	Mechanical	10000	5	Male	Rajkot
109	Krish	Mechanical	10000	7	Male	Baroda
110	Zeel	EE	9000	3	Female	Jamnagar

From the above given collection perform the following queries in MongoDB:

1. Display Female students.
2. Display students who belong to Rajkot city.
3. Display students studying in 7th sem.
4. Display students not studying in 3rd sem.
5. Display students whose roll no is greater than 107.
6. Display students whose city is Jamnagar or Baroda (use:IN)
7. Display students whose fees is less than 9000.
8. Display the roll no of those students who belongs to Mechanical department.
9. Display first 2 students names who lives in Baroda.
10. Display Male students who studying in 3rd sem.
11. Display sname and city and fees of those students whose roll no is less than 105.
12. Display documents where sname start with K.
13. Display documents where sname starts with Z or D in your collection.
14. Display documents where city starts with A to R in your collection.
15. Display students' info whose name start with P and ends in i.
16. Display students' info whose department name starts with 'C'.
17. Display name, sem, fees, and department whose city contains 'med' as three letters somewhere in city name.
18. Display name, sem, fees, and department who does not belongs to city Rajkot or Baroda.
19. Delete the documents whose city is Jamnagar.
20. Update sname of Krish to 'fenny' and gender to 'Female'.

Part – C:

1. Display next 2 students after skipping first 2 whose city is Ahmedabad.
2. Display rollno, sname, fees, and department of those students who is from Baroda and belongs to CE department.
3. Display documents where city name ends in 'oda'.
4. Display students' info whose name contains v. (Both uppercase(V) and lowercase(v))
5. Display students' info whose name starts with V and having 4 characters.

1

18 **Create collection as per following.**

DEPOSIT

ACTNO	CNAME	BNAME	AMOUNT	ADATE
101	ANIL	VRCE	1000.00	1-3-95
102	SUNIL	AJNI	5000.00	4-1-96
103	MEHUL	KAROLBAGH	3500.00	17-11-95
104	MADHURI	CHANDI	1200.00	17-12-95
105	PRMOD	M.G. ROAD	3000.00	27-3-96
106	SANDIP	ANDHERI	2000.00	31-3-96
107	SHIVANI	VIRAR	1000.00	5-9-95
108	KRANTI	NEHRU PLACE	5000.00	2-7-95
109	MINU	POWAI	7000.00	10-8-95

BRANCH

BNAME	CITY
VRCE	NAGPUR
AJNI	NAGPUR
KAROLBAGH	DELHI
CHANDI	DELHI
DHARAMPETH	NAGPUR
M.G. ROAD	BANGLORE
ANDHERI	BOMBAY
VIRAR	BOMBAY
NEHRU PLACE	DELHI
POWAI	BOMBAY

CUSTOMERS

CNAME	CITY
ANIL	CALCUTTA
SUNIL	DELHI
MEHUL	BARODA
MANDAR	PATNA
MADHURI	NAGPUR
PRAMOD	NAGPUR
SANDIP	SURAT
SHIVANI	BOMBAY
KRANTI	BOMBAY
NAREN	BOMBAY

BORROW

LOANNO	CNAME	BNAME	AMOUNT
201	ANIL	VRCE	1000.00
206	MEHUL	AJNI	5000.00
311	SUNIL	DHARAMPETH	3000.00
321	MADHURI	ANDHERI	2000.00
375	PRMOD	VIRAR	8000.00
481	KRANTI	NEHRU PLACE	3000.00

From the above given tables perform the following queries in MongoDB:
Part A:

1. Retrieve all data from table DEPOSIT.
2. Retrieve all data from table BORROW.
3. Retrieve all data from table CUSTOMERS.
4. Insert a record (550,'JAY','AJNI',NULL)in the BORROW table.
5. Display Account No, Customer Name & Amount from DEPOSIT.
6. Display Loan No, Amount from BORROW.
7. Display loan details of all customers who belongs to 'ANDHERI' branch.
8. Give account no and amount of depositor, whose account no is equals to 106.
9. Give name of borrowers having amount greater than 5000.
10. Give name of customers who opened account after date '1-12-96'.

Part B:

11. Display name of customers whose account no is less than 105.
12. Display name of customer who belongs to either 'NAGPUR' Or 'DELHI'.
13. Display name of customers with branch whose amount is greater than 4000 and account no is less than 105.
14. Find all borrowers whose amount is greater than equals to 3000 & less than equals to 8000.
15. Find all depositors who do not belongs to 'ANDHERI' branch.
16. Display Account No, Customer Name & Amount of such customers who belongs to 'AJNI', 'KAROLBAGH' Or 'M.G.ROAD' and Account No is less than 104.
17. Display all the details of first five customers.
18. Display all the details of first three depositors whose amount is greater than 1000.
19. Display Loan No, Customer Name of first five borrowers whose branch name does not belongs to 'ANDHERI'.
20. Retrieve all unique cities using DISTINCT. **(**Use **Customers collection)**

Part C:

21. Retrieve all unique branches using DISTINCT. **(**Use **Branch collection)**
22. Retrieve all the records of customer table as per their city name in ascending order.
23. Retrieve all the records of deposit table as per their amount column in descending order.
24. Update deposit amount of all customers from 3000 to 5000.
25. Change branch name of ANIL from VRCE to C.G. ROAD. (Use **Borrow collection**)
26. Update Account No of SANDIP to 111 & Amount to 5000.
27. Give 10% Increment in Loan Amount.
28. Update deposit amount of all depositors to 5000 whose account no between 103 & 107.
29. Update amount of loan no 321 to *NULL*.
30. Display the name of borrowers whose amount is *NULL*.

19 **Create collection as per following.**

STUDENT

RollNo	Name	Birthdate	SPI	City	Backlog	Branch
101	Keyur	5-1-92	8.5	Rajkot	2	CE
102	Hardik	15-2-94	9.0	Ahmedabad	0	CE
103	Kajal	14-3-96	10.00	Baroda	0	IT
104	Bhoomi	23-6-95	8.90	Ahmedabad	1	ICT
105	Harmit	15-2-94	9.80	Rajkot	1	IT
106	Jay	15-2-94	7.9	Rajkot	2	CE

From the above given tables perform the following queries in MongoDB:

Part A:

1. Give RollNo and Name of students, whose RollNo is greater than 103 and backlog is greater than 0 and branch is either CE or IT.
2. Give name of students whose SPI is between 8 and 9 and branch is either CE or IT.
3. Find all students who do not belongs to 'CE' branch.
4. Display RollNo and Name of first three students.
5. Display all the details of first three students whose SPI is greater than 8.5.
6. Retrieve all unique cities using DISTINCT.
7. Retrieve all unique branches using DISTINCT.
8. Retrieve all the records of student table as per their Backlog in descending order and then SPI in ascending order.
9. Update the branch and city of Jay to MCA and Jamangar respectively.
10. Update the backlog of Keyur and Bhoomi to *NULL*.

Part B:

1. Display the name of students whose backlog is *NULL* and backlog is greater than 1 and branch is either CE or IT.
2. Remove field Backlog for the students having 0 backlog.
3. Add new field mobile number in the Keyur's record with the value as '9825052365'
4. Remove birthdate field from all the documents.
5. Delete all the Employees who joined after 1-1-2007.
6. Delete all the records of Employee collection.

20 **Create collection as per following.**

EMPLOYEE

EmpNo	EmpName	JoiningDate	Salary	City
101	Keyur	5-1-02	12000.00	Rajkot
102	Hardik	15-2-04	14000.00	Ahmedabad
103	Kajal	14-3-06	15000.00	Baroda
104	Bhoomi	23-6-05	12500.00	Ahmedabad
102	Harmit	15-2-04	14000.00	Rajkot

From the above given tables perform the following queries in MongoDB:

Part A:

1. Display the name of employee whose salary is greater than 13000 and city is either Rajkot or Baroda.
2. Display the name of employee in ascending order by their name.
3. Retrieve all unique cities.
4. Update the city of Keyur and Bhoomi to *NULL*.
5. Display the name of employee whose city is *NULL*.
6. Delete all the records of Employee table having salary greater than and equals to 14000.
7. Delete all the Employees who belongs to 'RAJKOT' city.
8. Delete all the Employees who joined after 1-1-2007.
9. Delete all the records of Employee collection.
10. Remove Employee collection.

Part B:

ACCOUNT

ACTNO	CNAME	BNAME	AMOUNT	ADATE
101	ANIL	VRCE	1000.00	1-3-95
102	SUNIL	AJNI	5000.00	4-1-96
103	MEHUL	KAROLBAGH	3500.00	17-11-95
104	MADHURI	CHANDI	1200.00	17-12-95

105	PRMOD	M.G. ROAD	3000.00	27-3-96
106	SANDIP	ANDHERI	2000.00	31-3-96
107	SHIVANI	VIRAR	1000.00	5-9-95
108	KRANTI	NEHRU PLACE	5000.00	2-7-95
109	MINU	POWAI	7000.00	10-8-95

From the above given tables perform the following queries in MongoDB:

1. Retrieve all unique BNAME.
2. Display the Cname in ascending order by their amount and if amount is same then in descending order by cname.
3. Update the BNAME of Anil and Shivani to *NULL*.
4. Display the Cname of customers whose Bname is *NULL*.
5. Delete all the records of Account table having amount greater than and equals to 4000.
6. Delete all the accounts Bname is CHANDI.
7. Delete all the accounts having adate after 1-10-1995.
8. Delete all the records of Account collection.
9. Remove Account collection.

Part C:

ACCOUNT

ACTNO	CNAME	BNAME	AMOUNT	ADATE
101	ANIL	VRCE	1000.00	1-3-95
102	SUNIL	AJNI	5000.00	4-1-96
103	MEHUL	KAROLBAGH	3500.00	17-11-95
104	MADHURI	CHANDI	1200.00	17-12-95
105	PRMOD	M.G. ROAD	3000.00	27-3-96
106	SANDIP	ANDHERI	2000.00	31-3-96
107	SHIVANI	VIRAR	1000.00	5-9-95
108	KRANTI	NEHRU PLACE	5000.00	2-7-95
109	MINU	POWAI	7000.00	10-8-95

From the above given tables perform the following queries in MongoDB:

1. Display the Cname whose Bname is either AJNI or CHANDI and amount is greater than 3000 and sort the result in ascending order by their amount and if amount is same then in descending order by cname.
2. Retrieve top 3 unique BNAME and sort them in ascending order on BNAME.
3. Display the Cname whose ACTNO is greater than 103 and sort the result in ascending order by their amount and if amount is same then in descending order by cname.
4. Update the BNAME of Anil, Mehul and Shivani to *NULL*.
5. Display the Cname of customers whose Bname is *NULL*.
6. Update the amount of Anil to 5000.
7. Update amount of actno 109 to *NULL*.
8. Retrieve all the records of account table as per their bname in descending order.
9. Delete all the records of Account collection.
10. Remove Account collection.

21

From the above given collection perform the following queries in MongoDB:

Part – A: (Use EMPLOYEE collection of Lab No 17)

1. Display distinct city.
2. Display city wise number of persons.
3. Display sum of salary in your collection.
4. Display average of salary in your document.

5. Display maximum and minimum salary of your document.
6. Display city wise total salary in your collection.
7. Display gender wise maximum salary in your collection.
8. Display city wise maximum and minimum salary.
9. Display count of persons lives in Sydney city in your collection.
10. Display average salary of New York city.

Part – B: (Use STUDENT collection of Lab No 17)

1. Display distinct department.
2. Display city wise number of students.
3. Display sum of fees in your collection.
4. Display average of fees in your document.
5. Display maximum and minimum fees of your document.

Part – C: (Use STUDENT collection of Lab No 17)

1. Display department wise total fees in your collection.
2. Display gender wise maximum fees in your collection.
3. Display department maximum and minimum fees.
4. Display count of persons lives in Rajkot city in your collection.

Display department wise number of students.

22

Part A: Create collection as per following.

STUDENT

StuID	FirstName	LastName	Website	City	Division
1011	Keyur	Patel	techonthenet.com	Rajkot	II-BCX
1022	Hardik	Shah	digminecraft.com	Ahmedabad	I-BCY
1033	Kajal	Trivedi	bigactivities.com	Baroda	IV-DCX
1044	Bhoomi	Gajera	checkyourmath.com	Ahmedabad	III-DCW
1055	Harmit	Mitel	NULL	Rajkot	II-BCY
1066	Ashok	Jani	NULL	Baroda	II-BCZ

From the above given tables perform the following queries in MongoDB:

1. Display the name of students whose name starts with 'k'.
2. Display the name of students whose name consists of five characters.
3. Retrieve the first name & last name of students whose city name ends with a & contains six characters.
4. Display all the students whose last name ends with 'tel'.
5. Display all the students whose first name starts with 'ha' & ends with't'.
6. Display all the students whose first name starts with 'k' and third character is 'y'.
7. Display the name of students having no website and name consists of five characters.
8. Display all the students whose last name consist of 'jer'.
9. Display all the students whose city name starts with either 'r' or 'b'.
10. Display all the name students having websites.

Part B:

1. Display all the students whose name starts from alphabet A to H.
2. Display all the students whose name's second character is vowel.
3. Display student's name whose city name consist of 'rod'.
4. Retrieve the First & Last Name of students whose website name starts with 'bi'.
5. Display student's city whose last name consists of six characters.

Part C:

1. Display all the students whose city name consist of five characters & not starts with 'ba'.
2. Show all the student's whose division starts with 'II'.
3. Find out student's first name whose division contains 'bc' anywhere in division name.

4. Show student id and city name in which division consist of six characters and having website name.
5. Display all the students whose name's third character is consonant.

23

Part A: Create collection as per following.

CUSTOMER

CID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitucion 2222	Mexico D.F.	05021	Mexico
3	Antonio Moreno Taqueria	Antonio Moreno	Mataderos 2312	Mexico D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbkop	Christina Berglund	Berguvsvagen 8	Lulea	S-958 22	Sweden

From the above given tables perform the following queries in MongoDB:

1. Return all customers from a city that starts with 'L' followed by one wildcard character, then 'nd' and then two wildcard characters.
2. Return all customers from a city that contains the letter 'L'.
3. Return all customers from a city that do not contains the letter 'L'.
4. Return all customers that starts with 'La'.
5. Return all customers that do not starts with 'La'.
6. Return all customers that starts with Vowels.
7. Return all customers that do not starts with Vowels.
8. Return all customers that starts with 'a' to 'd' or 'x' to 'z'.
9. Return all customers that do not starts with 'a' to 'd' or 'x' to 'z'.
10. Return all customers that starts with Vowels as well as ends with Vowels.

Part B:

1. Return all customers that starts with 'a' or starts with 'b'.
2. Return all customers that starts with 'a' or starts with 'c' or starts with 't'.
3. Return all customers that starts with 'a' to 'd'.
4. Return all customers that ends with 'a'.
5. Return all customers that do not ends with 'a'.

Part C:

1. Return all customers that starts with 'b' and ends with 's'.
2. Return all customers that contains the phrase 'or'.
3. Return all customers that starts with "a" and are at least 3 characters in length.
4. Return all customers that have "r" in the second position.
5. Return all customers from Spain.

24

Part A: Create collection as per following.

EMPLOYEE

EID	EName	Department	Salary	JoiningDate	City
101	Rahul	Admin	56000	1-Jan-90	Rajkot
102	Hardik	IT	18000	25-Sep-90	Ahmedabad

103	Bhavin	HR	25000	14-May-91	Baroda
104	Bhoomi	Admin	39000	8-Feb-91	Rajkot
105	Rohit	IT	17000	23-Jul-90	Jamnagar
106	Priya	IT	9000	18-Oct-90	Ahmedabad
107	Neha	HR	34000	25-Dec-91	Rajkot

From the above given tables perform the following queries in MongoDB:

1. Display the Highest, Lowest, Total, and Average salary of all employees. Label the columns Maximum, Minimum, Total_Sal and Average_Sal, respectively.
2. Find total number of employees of EMPLOYEE table.
3. Give maximum salary from IT department.
4. Count total number of cities of employee without duplication.
5. Display city with the total number of employees belonging to each city.
6. Display city having more than one employee.
7. Give total salary of each department of EMPLOYEE table.
8. Give average salary of each department of EMPLOYEE table without displaying the respective department name.
9. Give minimum salary of employee who belongs to Ahmedabad.
10. List the departments having total salaries more than 50000 and located in city Rajkot.

Part B:

1. Count the number of employees living in Rajkot.
2. Display the difference between the highest and lowest salaries. Label the column DIFFERENCE.
3. Display the total number of employees hired before 1st January, 1991.
4. Display total salary of each department with total salary exceeding 35000 and sort the list by total salary.
5. List out department names in which more than two employees.
6. Return all employee whose name consist of 5 character and starts with 'a' or starts with 'b'.
7. Return all employee whose name consist of minimum 3 character and starts with 'b' or 'r' or 'p'.
8. Return all employee whose name ends with 'a' to 'd'.
9. Return all employee whose name ends with Vowels.
10. Return all employee whose name ends with Vowels or 'f' or 'g'.

Part C: Create collection as per following.

COMPANY

Title	Company	Type	Production_year	System	Production_cost	Revenue	Rating
Blasting Boxes	Simone Games	action adventure	1998	PC	100000	200000	7
Run Run Run!	13 Mad Bits	shooter	2011	PS3	3500000	650000	3
Duck n'Go	13 Mad Bits	shooter	2012	Xbox	3000000	1500000	5
SQL Wars!	Vertabelo	wargames	2017	Xbox	5000000	25000000	10
Tap Tap Hex!	PixelGaming Inc.	rhythm	2006	PS2	2500000	3500000	7
NoRisk	Simone Games	action adventure	2004	PS2	1400000	3400000	8

From the above given tables perform the following queries in MongoDB:

1. Display the name and total revenue for each company.
2. Generate a report with the production year and the number of games released this year (named count), the average of production cost for all games produced in this year (named avg_cost) and the average revenue for that year (named avg_revenue).
3. Count how many games of a given type are profitable (i.e. the revenue was greater than the production cost). Show the game type and the number of profitable games (named number_of_games) for each type.
4. Obtain the type of games and the total revenue generated for games with a production_year after 2010 and with a PS2 or PS3 system. Order the result so the types with the highest revenue come first.
5. For all companies present in the table, obtain their names and the sum of gross profit over all years. (Assume that gross profit = revenue - cost of production). Name this column gross_profit_sum. Order the results by gross profit, in descending order.
6. Obtain the yearly gross profit of each company. In other words, we want a report with the company name, the year, and the gross profit for that year. Order the report by company name and year.
7. For each company, select its name, the number of games it's produced (as the number_of_games column), and the average cost of production (as the avg_cost column). Show only companies producing more than one game.

25 **Create collection as per following.**

STUDENT

Rno	Name	Branch
101	Raju	CE
102	Amit	CE
103	Sanjay	ME
104	Neha	EC
105	Meera	EE
106	Mahesh	ME

RESULT

Rno	SPI
101	8.8
102	9.2
103	7.6
104	8.2
105	7.0
107	8.9

EMPLOYEE

EmployeeNo	Name	ManagerNo
E01	Tarun	*NULL*
E02	Rohan	E02
E03	Priya	E01
E04	Milan	E03
E05	Jay	E01
E06	Anjana	E04

From the above given tables perform the following queries in MongoDB:

Part A:

1. Combine information from student and result table using cross join or Cartesian product.
2. Display Rno, Name, Branch and SPI of all students.
3. Display Rno, Name, Branch and SPI of CE branch's student only.
4. Display Rno, Name, Branch and SPI of other than EC branch's student only.
5. Display average result of each branch.
6. Display average result of each branch and sort them in ascending order by SPI.
7. Display average result of CE and ME branch.
8. Perform the left outer join on Student and Result tables.
9. Perform the right outer join on Student and Result tables.
10. Perform the full outer join on Student and Result tables.
11. Retrieve the names of employee along with their manager name from the Employee table.

Part-B:

Create collection as per following.

PERSON

PersonID	PersonName	DepartmentID	Salary	JoiningDate	City
101	Rahul Tripathi	2	56000	01-01-2000	Rajkot
102	Hardik Pandya	3	18000	25-09-2001	Ahmedabad
103	Bhavin Kanani	4	25000	14-05-2000	Baroda

104	Bhoomi Vaishnav	1	39000	08-02-2005	Rajkot
105	Rohit Topiya	2	17000	23-07-2001	Jamnagar
106	Priya Menpara	NULL	9000	18-10-2000	Ahmedabad
107	Neha Sharma	2	34000	25-12-2002	Rajkot
108	Nayan Goswami	3	25000	01-07-2001	Rajkot
109	Mehul Bhundiya	4	13500	09-01-2005	Baroda
110	Mohit Maru	5	14000	25-05-2000	Jamnagar

DEPARTMENT

DepartmentID	DepartmentName	DepartmentCode	Location
1	Admin	Adm	A-Block
2	Computer	CE	C-Block
3	Civil	CI	G-Block
4	Electrical	EE	E-Block
5	Mechanical	ME	B-Block

From the above given table perform the following queries in MongoDB:

1. Find all persons with their department name & code.
2. Give department wise maximum & minimum salary with department name.
3. Find all departments whose total salary is exceeding 100000.
4. Retrieve person name, salary & department name who belongs to Jamnagar city.
5. Find all persons who does not belongs to any department.
6. Find department wise person counts.
7. Find average salary of person who belongs to Ahmedabad city.
8. Produce Output Like: <PersonName> earns <Salary> from department <DepartmentName> monthly. (In Single Column)
9. List all departments who have no persons.
10. Find city & department wise total, average & maximum salaries.

Part - C

1. Display Unique city names.
2. List out department names in which more than two persons.
3. Combine person name's first three characters with city name's last three characters in single column.
4. Give 10% increment in Computer department employee's salary.
5. Display all the person name's who's joining dates difference with current date is more than 365 days.

26 **Part A**: Create database in MongoDB for the following RDBMS schema and enter data given in below tables.

STUDENT

Rno	Name	City	DID
101	Raju	Rajkot	10
102	Amit	Ahmedabad	20
103	Sanjay	Baroda	40
104	Neha	Rajkot	20
105	Meera	Ahmedabad	30
106	Mahesh	Baroda	10

ACADEMIC

Rno	SPI	Bklog
101	8.8	0
102	9.2	2
103	7.6	1
104	8.2	4
105	7.0	2
106	8.9	3

DEPARTMENT

DID	DName
10	Computer
20	Electrical
30	Mechanical
40	Civil

Part B: Update the below records in above MongoDB database.

Subject : 2305CS101 – Database Management Systems

2	<table><tr><th>Rno</th><th>Mobile</th></tr><tr><td>101</td><td>12345678</td></tr><tr><td>101</td><td>23456789</td></tr><tr><td>103</td><td>23456781</td></tr><tr><td>103</td><td>52345678</td></tr><tr><td>103</td><td>82345678</td></tr><tr><td>103</td><td>42345678</td></tr></table> **Part C**: 1. Delate any one mobile no of student whose Rno is 103. 2. Update any one mobile no of student whose Rno is 103. 3. Delete mobile no field of student whose Rno is 101. 4. Update address of student whose Rno is 105 as (Building Name: 'Darshan Building', Road Name: 'Raiya Road', Area: 'KKV area') 5. Delete all the documents of Computer Department.