**INTERVIEWS**

*A*

***Project Report***

*submitted in partial fulfillment of the*

*requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE & ENGINEERING**

Specialization

**Business Analytics and Optimization**

by

| **Student Name** | **Roll Number** |
| --- | --- |
| **Kshitij Priyadarshi** | **R103218071** |
| **Nandini Batra** | **R103218039** |
| **Mridul Jaiswal** | **R103218046** |
| **Jay Sharma** | **R103218065** |

*under the guidance of*

**Ms. Roohi Sille**

upes-new-logo

**Department of Informatics**

**School of Computer Science**

**University of Petroleum & Energy Studies**

**Bidholi, Via Prem Nagar, Dehradun, UK**

**May – 2021**

upes-new-logo

**DECLARATION**

I/We hereby certify that the project work entitled **“INTERVIEW”** in partial fulfilment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING specialization **Business Analytics and Optimization**and submitted to the Department of Informatics, School of Computer Science, University of Petroleum & Energy Studies, Dehradun, is an authentic record of my/ our work carried out during a period from **June 2021** to **July 2021** under the supervision of **Ms Roohi Sille**.

The matter presented in this project has not been submitted by me/ us for the award of any other degree of this or any other University.

|  | **Kshitij Priyadarshi** | **Nandini Batra** | **Mridul Jaiswal** | **Jay Sharma** |
| --- | --- | --- | --- | --- |
|  | **R103218071** | **R103218039** | **R103218046** | **R103218065** |

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

**Date: 11/08/2021**  **Roohi Sille**

Project Mentor

**Dr. Priyadarshan Patra**

Head**,** Department of Informatics

School of Computer Science

University of Petroleum & Energy Studies

Dehradun – 248 001 (Uttarakhand)

upes-new-logo

**ACKNOWLEDGEMENT**

We wish to express our deep gratitude to our mentor(s) **Roohi Silles**, for all advice, encouragement and constant support he has given us throughout our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our respected Head of Department, Informatics, **Dr. Priyadarshan Patra,** for his guidance and support as and when required.

We are also grateful to **Dr. Priyadarshan Patra, Dean SCS**, for providing the necessary facilities to carry out our project work successfully.

We would like to thank all our friends for their help and constructive criticism during our project work. Finally, we have no words to express our sincere gratitude to our **parents** who have shown us this world and for everything they have given to us.

|  | **Kshitij Priyadarshi** | **Nandini Batra** | **Mridul Jaiswal** | **Jay Sharma** |
| --- | --- | --- | --- | --- |
|  | **R103218071** | **R103218039** | **R103218046** | **R103218065** |

**INTRODUCTION**

**SQL** is a [domain-specific language](https://en.wikipedia.org/wiki/Domain-specific_language) used in programming and designed for managing data held in a [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS), or for stream processing in a [relational data stream management system](https://en.wikipedia.org/wiki/Relational_data_stream_management_system) (RDSMS). It is particularly useful in handling [structured data](https://en.wikipedia.org/wiki/Data_model), i.e. data incorporating relations among entities and variables.

SQL offers two main advantages over older read-write [APIs](https://en.wikipedia.org/wiki/API) such as [ISAM](https://en.wikipedia.org/wiki/ISAM) or [VSAM](https://en.wikipedia.org/wiki/VSAM). Firstly, it introduced the concept of accessing many records with one single command. Secondly, it eliminates the need to specify *how* to reach a record, e.g. with or without an [index](https://en.wikipedia.org/wiki/Database_index).

Originally based upon [relational algebra](https://en.wikipedia.org/wiki/Relational_algebra) and [tuple relational calculus](https://en.wikipedia.org/wiki/Tuple_relational_calculus), SQL consists of many types of statements which may be informally classed as [sublanguages](https://en.wikipedia.org/wiki/Sublanguage), commonly: a [data query language](https://en.wikipedia.org/wiki/Data_query_language) (DQL), a [data definition language](https://en.wikipedia.org/wiki/Data_definition_language) (DDL), a [data control language](https://en.wikipedia.org/wiki/Data_control_language) (DCL), and a [data manipulation language](https://en.wikipedia.org/wiki/Data_manipulation_language) (DML). The scope of SQL includes data query, data manipulation (insert, update and delete), data definition ([schema](https://en.wikipedia.org/wiki/Database_schema) creation and modification), and data access control. Although SQL is essentially a [declarative language](https://en.wikipedia.org/wiki/Declarative_programming) ([4GL](https://en.wikipedia.org/wiki/4GL)), it also includes [procedural](https://en.wikipedia.org/wiki/Procedural_programming) elements.

SQL was one of the first commercial languages to use [Edgar F. Codd](https://en.wikipedia.org/wiki/Edgar_F._Codd)’s [relational model](https://en.wikipedia.org/wiki/Relational_model). The model was described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks".[[8]](https://en.wikipedia.org/wiki/SQL#cite_note-codd-relational-model-11) Despite not entirely adhering to [the relational model as described by Codd](https://en.wikipedia.org/wiki/Codd%27s_12_rules), it became the most widely used database language

**Problem Statement**

* Basics of SQL - Joins, views, stored procedures, CTEs, sub queries etc.
* In SQL DWH/Database how to apply while loop in the procedure and also have knowledge about the triggers
* How to use casing condition in join with implementation

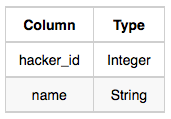
**Task 1:-**

Julia conducted a days of learning SQL contest. The start date of the contest was *March 01, 2016*, and the end date was *March 15, 2016*. Write a query to print a total number of unique hackers who made at least submission each day (starting on the first day of the contest), and find the *hacker\_id* and *name* of the hacker who made a maximum number of submissions each day. If more than one such hacker has a maximum number of submissions, print the lowest *hacker\_id*. The query should print this information for each day of the contest, sorted by the date**.**

**Input Format**

The following tables hold contest data:

* *Hackers:* The *hacker\_id* is the id of the hacker, and *name* is the name of the hacker.
* *Submissions:* The *submission\_date* is the date of the submission, *submission\_id* is the id of the submission, *hacker\_id* is the id of the hacker who made the submission, and *score* is the score of the submission.



**Task 2:-**

Consider P1(a,b) and P2(c,d) to be two points on a *2D* plane.

* happens to equal the minimum value in *Northern Latitude* (*LAT\_N* in STATION).
* happens to equal the minimum value in *Western Longitude* (*LONG\_W* in STATION).
* happens to equal the maximum value in *Northern Latitude* (*LAT\_N* in STATION).
* happens to equal the maximum value in *Western Longitude* (*LONG\_W* in STATION).

Query the [Manhattan Distance](https://xlinux.nist.gov/dads/HTML/manhattanDistance.html) between points P1 and P2 and round it to a scale of decimal places.

**Input Format**

The STATION table is described as follows:

*LAT\_N* is the northern latitude and *LONG\_W* is the western longitude.

. Write a query to print all *prime numbers* less than or equal to 1000. Print your result on a single line, and use the ampersand (&) character as your separator (instead of a space).

**Task 3:-**

[Pivot](https://en.wikipedia.org/wiki/Pivot_table) the *Occupation* column in OCCUPATIONS so that each *Name* is sorted alphabetically and displayed underneath its corresponding *Occupation*. The output column headers should be *Doctor*, *Professor*, *Singer*, and *Actor*, respectively.

Note: Print NULL when there are no more names corresponding to an occupation.

**Input Format:-** The OCCUPATIONS table is described as follows: *Occupation* will only contain one of the following values: Doctor, Professor, Singer or Actor.

**Task 4:-**

You are given a table, *BST*, containing two columns: *N* and *P,* where *N* represents the value of a node in *Binary Tree*, and *P* is the parent of *N*.

Write a query to find the node type of *Binary Tree* ordered by the value of the node. Output one of the following for each node:

* *Root*: If the node is the root node.
* *Leaf*: If node is a leaf node.
* *Inner*: If the node is neither root nor leaf node.

**Task 5:-**

Amber's conglomerate corporation just acquired some new companies. Each of the companies follows this hierarchy:

Given the table schemas below, write a query to print the *company\_code*, *founder* name, the total number of *lead* managers, the total number of *senior* managers, the total number of *managers*, and the total number of *employees*. Order your output by ascending *company\_code*.

Note:

* The tables may contain duplicate records.
* The *company\_code* is a string, so the sorting should not be numeric. For example, if the *company\_codes* are *C\_1*, *C\_2*, and *C\_10*, then the ascending *company\_codes* will be *C\_1*, *C\_10*, and *C\_2*.

**Input Format**

The following tables contain company data:

* *Company:* The *company\_code* is the code of the company and *the founder* is the founder of the company.
* *Lead\_Manager:* The *lead\_manager\_code* is the code of the lead manager, and the *company\_code* is the code of the working company.
* *Senior\_Manager:* The *senior\_manager\_code* is the code of the senior manager, the *lead\_manager\_code* is the code of its lead manager, and the *company\_code* is the code of the working company.
* *Manager:* The *manager\_code* is the code of the manager, the *senior\_manager\_code* is the code of its senior manager, the *lead\_manager\_code* is the code of its lead manager, and the *company\_code* is the code of the working company.
* *Employee:* The *employee\_code* is the code of the employee, the *manager\_code* is the code of its manager, the *senior\_manager\_code* is the code of its senior manager, the *lead\_manager\_code* is the code of its lead manager, and the *company\_code* is the code of the working company.

**Task 6:-**

You are given three tables: *Students*, *Friends* and *Packages.* *Students* contain two columns: *ID* and *Name*. *Friends* contain two columns: *ID* and *Friend\_ID* (*ID* of the ONLY best friend). *Packages* contain two columns: *ID* and *Salary* (offered salary in $ thousands per month).

Write a query to output the names of those students whose best friends got offered a higher salary than them. Names must be ordered by the salary amount offered to the best friends. It is guaranteed that no two students got the same salary offer.

**Methodology**

1. The Projet had 6 tasks in total which were disturbed amongst 4 of us mutually.
2. The tasks consisted of the creation of a dataset, various join operations, nested and subqueries.
3. We created an arbitrary dataset to perform the SQL queries.
4. Combining all the queries of a specific task output was generated.