A report on

SATEUREKA – THE MODERN-DAY SATELLITE DATABASE

by

2018103547 | S. KARTHICK RAJA

2018103553 | KIRAN SEKHAR

2018103601 | S. SRIHARI

Submitted for the course

CS6105- Database Management Systems

Evaluator Name and Signature

Date:

Abstract

Assembled by experts at the Union of Concerned Scientists (UCS), the **SATEUREKA** is a listing of many operational satellites orbiting around Earth. Our intent in producing the database is to create a research tool for specialists and non-specialists alike by collecting open-source information on operational satellites and presenting it in a format that can be easily manipulated for research and analysis.

SATEUREKA denotes “SATELLITE - EUREKA”, which stores golden significant information about the rocketry system across globe. This database minimizes redundancy of data, and in the same time provides the required information at our doorstep.

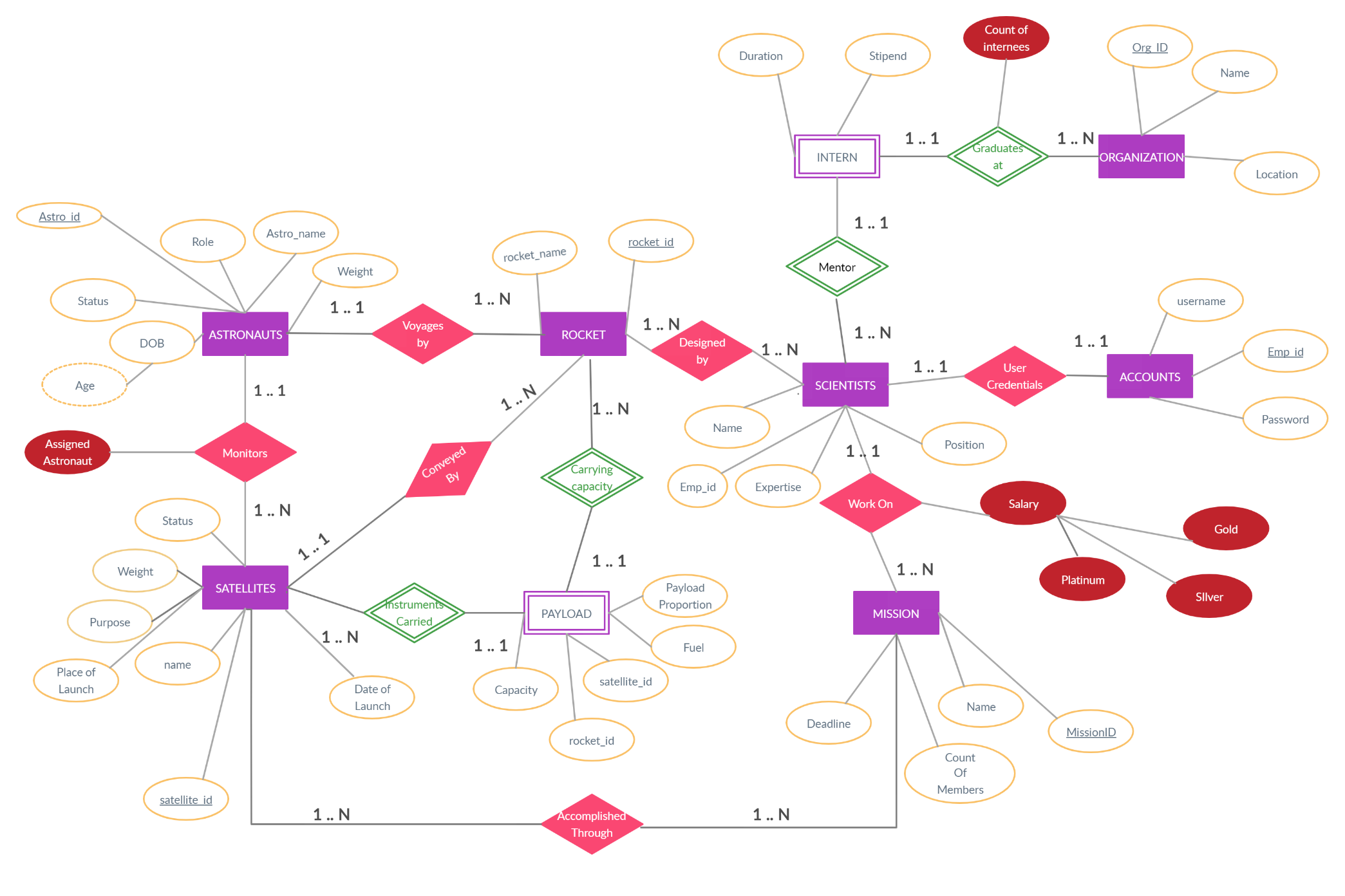
Privilege of dashboard access is given only to the project directors who can perform a variety of operations in the dashboard. It includes viewing the details of candidates who would be interning at the organization. Upload of Fuel amount and getting the value of payload proportions are done with the help of stored procedures and functions. Triggers are used to indicate a fuel warning if the fuel capacity drops drastically. Backup information of the employees who have retired or resigned is achieved using triggers. Views are used to get the list of active satellites and the astronaut list who are fit to pursue their missions.

Front End: HTML, CSS, JAVASCRIPT

Back End: PHP, MYSQL

Salient Features of the System

1. The Mentors List for internees would be available in the home page, along with their expertise which enables the interns to choose the scientist under whom they wish to work based on their domain of interest.
2. Confidential status of Missions can be obtained.
3. An auto warn system to indicate Fuel-Warningif the fuel amount of the rocket falls below 50% mark.
4. Authenticated information storage of scientists who have resigned or retired.
5. Provision of quick retrieval of Astronaut details, including their fitness, role in the mission, exclusively for Project Directors.
6. Three level monitored scrutinizing system for viewing the dashboard.
7. A chat query system which addresses the queries raised directly by the project director.
8. Countdown for the upcoming mission is always displayed in the homepage.
9. Latest Achievements would be displayed



SATEUREKA

ER Diagram

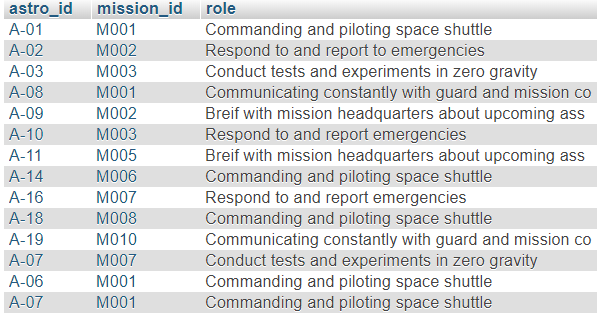
DATABASE Schema

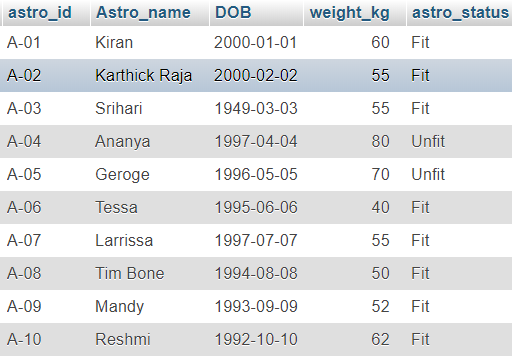
SATEUREKA

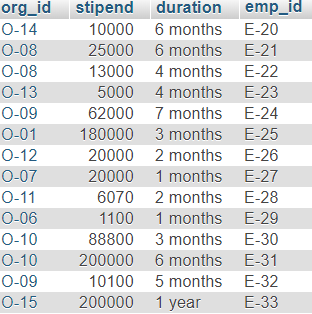
A sign on the side of a building

Description automatically generated

Instances of each Relation

ASSIGNED\_ASTRO 

ASTRONAUTS 

INTERN 

MISSION A screenshot of a cell phone

Description automatically generated

ORGANISATION A screenshot of a cell phone

Description automatically generated

PAYLOAD A picture containing people, holding, group, man

Description automatically generated

ROCKET A screenshot of a cell phone

Description automatically generated

SALARY\_INFO A picture containing woman, table, wooden, hanging

Description automatically generated

SATELLITES

A screenshot of a cell phone

Description automatically generated

SCIENTISTS

A screenshot of a cell phone

Description automatically generated

Views

|  |  |  |  |
| --- | --- | --- | --- |
| # | **Name** | **Tables Involved** | **Description** |
| 1 | Active – Satellites | Satellites | To indicate the list of satellites  Which are currently active. |
| 2 | Fit- Astronauts | Astronauts | To provide the essential details of the astronaut who are fit enough to get a GO for their mission |
| 3 | Internship | Intern, Scientists,  organisation | To display the list of internees who have been qualified to work in Sateureka |
| 4 | Mentors | Scientists | To provide the list of mentors of Sateureka, so that qualified interns can approach them based on their expertise |

Procedures

|  |  |  |  |
| --- | --- | --- | --- |
| # | **Name & Description** | **Tables Involved** | **IN & OUT Parameter(s)** |
| 1 | **Get Scientist Level**  A procedure to display earnings of all scientists, using a categorization level based on their salary | Salary info | **IN** – EmployeeNumber VARCHAR 4 |
| **OUT** – EmployeeLevel VARCHAR 20 |
| 2 | **Mission Name**  A procedure to provide mission name and satellite name of the scientist given his employee id. | Scientists | **IN** – EmpID VARCHAR 4  **OUT** – MissionName VARCHAR 20  **OUT** – SatelliteName VARCHAR 20 |
| 3 | **Get\_Scientist\_Profile**  A procedure to display the position, expertise and the mission, satellite under which the scientist is currently working | Scientist, Mission, Satellites | **IN** – EmpId VARCHAR 4  **OUT**- Position Varchar 80  **OUT**- Expertise Varchar 80  **OUT**- MissionName Varchar 80  **OUT**- SatelliteName Varchar 80 |
| 4 | **List\_Name**  A procedure which lists the names of internees representing a geographical location | Organisation, Intern, Satellites | **IN** – Location Varchar 4  **INOUT** – Namelist Varchar 400 |

Functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Name & Description** | **Tables Involved** | **Parameter(s)** | **Return Type** |
| 1 | **Mission\_status**  a function that gets the mission\_id and find the corresponding mission status | Mission, Satellites | (mid VARCHAR (4)) | varchar |
| 2 | **Payload\_proportion (in percentage)**  A function that gets the satellite\_id and returns the corresponding payload proportion in percentage | Payload | (sid VARCHAR (4)) | float |
| 3 | **Salary (In lakhs)**  A function that gets the emp\_id and returns his/her salary figures in Lakhs | Salary\_info | (eid VARCHAR (4)) | decimal (8,4) |
| 4 | **get\_astronaut\_role**  A function that returns the role of an astronaut given his id | Astronauts,  Assigned\_astro | (inpname VARCHAR (20)) | varchar (60) |

Triggers

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name & Description** | **Trigger Type** | **Tables Involved** |
| 1 | **Astro\_status**  Trigger is to set the status of a newly enrolled astronaut as FIT. | Before Insert | Assigned\_astro,  astronauts |
| 2 | **Fuel\_status**  Trigger is to set the satellite status as FUEL WARNING or ACTIVE based on the updated fuel amount | After Update | Payload, Satellites |
| 3 | **Intern\_checker**  Trigger is to increment the count based on new internees | Before Insert | Intern, Organisation\_count |
| 4 | **Intern\_dechecker**  Trigger is to maintain the organization count of internees and to delete from Employees table of Sateureka | Before Delete | Intern, organization\_count, scientist |
| 5 | **Resigned\_or\_retire**  Trigger is to backup essential details of outgoing employees and to delete their salary info | Before delete | Scientists, Salary\_info |

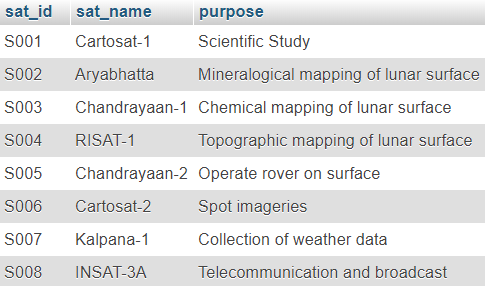
**SQL**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Tables Involved** | **Constructs Used** |
| 1 | To provide the details of internees grouped based on organization name and duration of the internship | Intern, Organization | Aggregate Function, Join, Group by |
| 2 | To provide an in-depth insight about missions and the corresponding satellites involved | Mission, Satellites | Join |
| 3 | To find the most successful mission in terms of salary earned by grouping the salary of members in the same mission | Scientists, Salary\_info | Aggregate function, Join, Group by, Order by |
| 4 | To find the name and emp\_id of scientists who are not interns | Scientists, Intern | Sub queries |
| 5 | To find the best rockets in terms of their capacity and fuel amount | Rocket, Payload | Join, Order by |
| 6 | To display the details of missions which are on time. i.e. Launched before their deadline | Satellites, Mission | Join, Date functions |
| 7 | To provide the astronaut profile in his/her mission | Astronauts, Assigned\_astro, Mission | Join, Order by |

***VIEWS***

1. **create view `active - satellites` as select sat\_id,sat\_name, purpose from `satellites`;**
2. **create view `fit-astronauts` as select astro\_id,Astro\_name,weight\_kg from `astronauts`;**
3. **create view `internship` as select emp\_id,ename,org\_id,name,duration from `organisation` natural join `intern` natural join `scientists`;**
4. **create view `mentor` as select emp\_id,ename,expertise from `scientists`;**

***VIEWS***

**ACTIVE SATELLITES**  

***FIT ASTRONAUTS***  A screenshot of a cell phone

Description automatically generated

***INTERNSHIP***

A screenshot of a cell phone

Description automatically generated

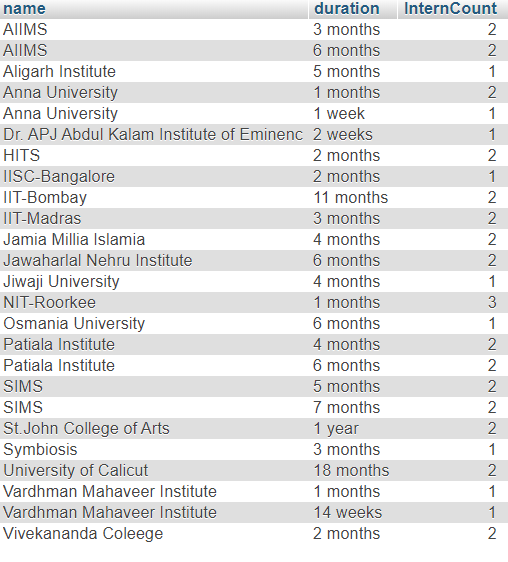
***MENTORS***

A screenshot of a cell phone

Description automatically generated

***SQL QUERIES***

1. **SELECT name, duration,count(emp\_id) as InternCount FROM `intern` natural join `organisation` group by `name`,`duration`;**



1. **SELECT \* FROM `mission` NATURAL JOIN `satellites`;**A screenshot of a cell phone

   Description automatically generated
2. **SELECT mission\_id, sum(salary) FROM `scientists` NATURAL JOIN `salary\_info` group by mission\_id order by sum(salary);**

A screenshot of a cell phone

Description automatically generated

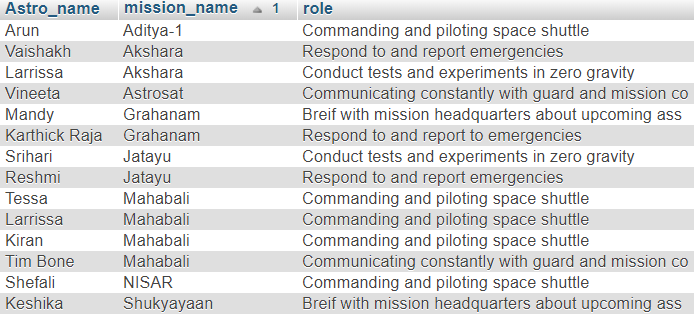
1. **select a.emp\_id,a.ename from `scientists` as a where a.emp\_id NOT IN (select b.emp\_id from `intern` as b );**

A screenshot of a cell phone

Description automatically generated

1. **SELECT rocketname, capacity\_tons,fuel FROM `rocket` NATURAL JOIN `payload` order by capacity\_tons desc,fuel desc;A screenshot of a cell phone

   Description automatically generated**
2. **select \* from `satellites` natural join `mission` where DATEDIFF(DOL, deadline) < 0;**A screenshot of a cell phone

   Description automatically generated
3. **select Astro\_name,mission\_name,role from `astronauts` NATURAL join `assigned\_astro` natural join `mission` order by mission\_name;**

***PROCEDURES***

1. **GetScientistLevel**

CREATE PROCEDURE `GetScientistLevel`(IN `EmployeeNumber` VARCHAR (4), OUT `EmployeeLevel` VARCHAR(20))

BEGIN

DECLARE credit DECIMAL (10,2) ;

SELECT salary INTO credit FROM salary\_info WHERE emp\_id = EmployeeNumber;

IF(credit > 50000 )THEN

SET EmployeeLevel = 'PLATINUM';

ELSEIF (credit <= 50000 AND credit > 10000) THEN

SET EmployeeLevel = 'GOLD';

ELSE

SET EmployeeLevel = 'SILVER';

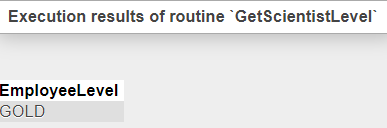
END IF;

END

SET @P0=’E-81’

CALL `GetScientistLevel` (@p0,@p1);

SELECT @p1 AS `EmployeeLevel`;



1. **MissionName**

CREATE PROCEDURE `MissionName`(IN `Empid` VARCHAR (4), OUT `M\_Name` VARCHAR(20), OUT `S\_Name` VARCHAR(20))

BEGIN

DECLARE mid VARCHAR (100);

SELECT mission\_id INTO mid FROM scientists WHERE emp\_id = Empid;

CASE mid

WHEN 'M001' THEN

SET M\_Name = 'Mahabali';

SET S\_Name = 'Ladee';

WHEN 'M002' THEN

SET M\_Name = 'Grahanam';

SET S\_Name = 'Chandrayaan-2';

WHEN 'M003' THEN

SET M\_Name = 'Jatayu';

SET S\_Name = 'Maven';

WHEN 'M004' THEN

SET M\_Name = 'Mars Orbiter Mission';

SET S\_Name = 'Mangalyaan';

WHEN 'M005' THEN

SET M\_Name = 'Shukyayaan';

SET S\_Name = 'Saral';

WHEN 'M006' THEN

SET M\_Name = 'Aditya-1';

SET S\_Name = 'Galex';

WHEN 'M007' THEN

SET M\_Name = 'Akshara';

SET S\_Name = 'Cartosat-2';

WHEN 'M008' THEN

SET M\_Name = 'NISAR';

SET S\_Name = 'INSAT-3A';

WHEN 'M009' THEN

SET M\_Name = 'Gaganyaan';

SET S\_Name = 'Bhaskara';

WHEN 'M010' THEN

SET M\_Name = 'Astrosat';

SET S\_Name = 'Mariner';

ELSE

SET M\_Name = 'Enter a valid mission no';

SET S\_Name = ' ';

END CASE;

END

SET @p0=’E-99’;

CALL `MissionName`(@p0,@p1,@p2);

SELECT @p1 AS `M\_Name`,@p2 AS `S\_Name`;

A screenshot of a cell phone

Description automatically generated

1. **get\_scientist\_profile**

CREATE PROCEDURE `get\_scientist\_profile`(IN `names` VARCHAR(40), OUT `de\_pos` VARCHAR(40), OUT `de\_exp` VARCHAR(40), OUT `de\_sat` VARCHAR(40), OUT `de\_m` VARCHAR(40))

BEGIN

DECLARE de\_names VARCHAR( 40 ) DEFAULT "";

DECLARE de\_nams VARCHAR( 40 ) DEFAULT "";

DECLARE de\_na VARCHAR( 40 ) DEFAULT "";

DECLARE de\_n VARCHAR( 40 ) DEFAULT "";

SELECT POSITION INTO de\_names FROM scientists WHERE emp\_id = names;

SELECT EXPERTISE INTO de\_nams FROM scientists WHERE emp\_id = names;

SELECT sat\_name INTO de\_na FROM scientists NATURAL JOIN mission NATURAL JOIN satellites WHERE emp\_id = names;

SELECT mission\_name INTO de\_n FROM scientists NATURAL JOIN mission NATURAL JOIN satellites WHERE emp\_id = names;

SET de\_exp = de\_nams;

SET de\_pos = de\_names;

SET de\_sat = de\_na;

SET de\_m = de\_n;

END

SET @p0=’E-61’;

CALL `get\_scientist\_profile`(@p0,@p1,@p2,@p3,@p4);

SELECT @p1 AS `de\_pos` , @p2 AS `de\_exp` , @p3 AS `de\_sat` , @p4 AS `de\_m`;



**4. list\_name**

CREATE PROCEDURE `list\_name`(IN `loc` VARCHAR(20), INOUT `name\_list` VARCHAR(400))

BEGIN

DECLARE is\_done INTEGER DEFAULT 0;

DECLARE s\_name varchar(200) DEFAULT "";

DECLARE stud\_cursor CURSOR FOR

SELECT ename FROM organisation NATURAL JOIN intern NATURAL JOIN scientists WHERE location = loc;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET is\_done = 1;

OPEN stud\_cursor;

get\_list : LOOP

FETCH stud\_cursor INTO s\_name;

IF is\_done = 1 THEN

LEAVE get\_list;

END IF;

SET name\_list = CONCAT(s\_name,' ;', name\_list);

END LOOP get\_list;

CLOSE stud\_cursor;

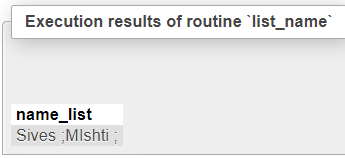
END

SET @p0 = ‘Mumbai’;

SET @p1=’’;

CALL `list\_name`(@p0,@p1);

SELECT @p1 AS `name\_list`;



**FUNCTIONS**

1. **Mission\_Status**

CREATE FUNCTION `Mission\_Status`(`mid` VARCHAR(4))

RETURNS VARCHAR(20) CHARSET

BEGIN

DECLARE C VARCHAR(20) ;

SELECT sat\_status into C from `mission` natural join `satellites` where mission\_id = mid;

RETURN C;

END

SET @p0=’M010’;

SELECT `MissionStatus`(@p0) AS `MissionStatus`;

A screenshot of a social media post

Description automatically generated

1. **Payload\_proportion(in percentage)**

CREATE FUNCTION `Payload\_proportion(in percentage)`(`s` VARCHAR(4))

RETURNS FLOAT

BEGIN

DECLARE C NUMERIC(6,6) ;

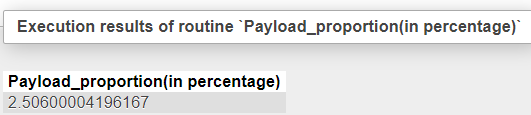
SELECT payload\_prop into C from `payload` where sat\_id = s;

RETURN C\*100;

END

SET @p0=’S020’;

SELECT `Payload\_Proportion(in percentage)`(@p0) AS ` Payload\_Proportion(in percentage)`;



1. **Salary(In lakhs)**

CREATE FUNCTION `Salary(In lakhs)`(`e` VARCHAR(4))

RETURNS DECIMAL(8,4)

BEGIN

DECLARE C INT ;

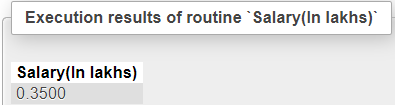
SELECT salary into C from `salary\_info` where emp\_id = e;

RETURN C/100000;

END

SET @p0=’E-36’;

SELECT `Salary(In lakhs)`(@p0) AS `Salary(In lakhs)`;



1. **get\_astronaut\_role**

CREATE FUNCTION `get\_astronaut\_role`(`inpname` VARCHAR(20))

RETURNS VARCHAR(60) CHARSET

BEGIN

DECLARE de\_name VARCHAR( 60 ) DEFAULT "";

SELECT role INTO de\_name FROM astronauts NATURAL JOIN assigned\_astro WHERE Astro\_name = inpname;

RETURN de\_name;

END

SET @p0=’Shefali’;

SELECT `get\_astronaut\_role` (@p0) AS `get\_astronaut\_role`;

