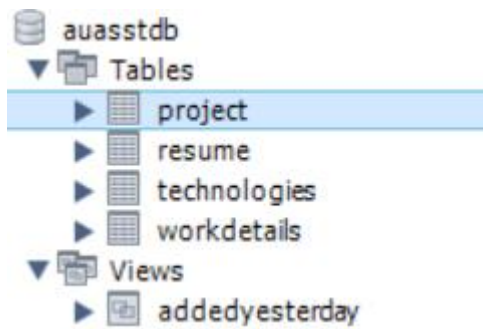


## TABLES:



## TABLE SCHEMAS:

### PROJECT:

#### Table: **project**

##### Columns:

<b>id</b>	int(11) AI PK
<b>resume_id</b>	int(11)
proj_name	varchar(50)
proj_description	varchar(100)
github_link	varchar(100)
technologies_used	varchar(150)

(resume\_id foreign key references pk of resume table)

### RESUME:

Table: <b>resume</b>	
Columns:	
<b>id</b>	int(11) AI PK
person_name	varchar(100)
email	varchar(100)
phone	varchar(15)
address	varchar(300)
location	varchar(50)
objective	varchar(500)
hobbies	varchar(500)
certifications	varchar(500)
added_date	date

(resume\_id foreign key references pk of resume table)

#### WORK DETAILS:

Table: **workdetails**

Columns:

<b>id</b>	int(11) AI PK
<b>resume_id</b>	int(11)
company_name	varchar(50)
duration	varchar(50)
work_description	varchar(50)

(resume\_id foreign key references pk of resume table)

#### TECHNOLOGIES:

Table: **technologies**

Columns:

<b><u>tech_name</u></b>	varchar(50) PK
<b><u>resume_id</u></b>	int(11) PK

(resume\_id foreign key references pk of resume table)

## 1. CREATING TABLES

The screenshot shows a SQL IDE interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including tables like 'project', 'resume', 'technologies', and 'workdetails'. The main editor displays SQL queries for creating tables: 'WorkDetails' and 'Technologies'. The 'Output' pane at the bottom shows the execution results, including the time, action, and message for each query.

```
26 FOREIGN KEY (resume_id) REFERENCES Resume (id) ON DELETE CASCADE
27 );
28
29 CREATE TABLE WorkDetails(
30     id int NOT NULL auto_increment,
31     resume_id int NOT NULL,
32     company_name varchar(50) NOT NULL,
33     duration varchar(50) NOT NULL,
34     work_description varchar(50),
35     PRIMARY KEY (id),
36     FOREIGN KEY (resume_id) REFERENCES Resume (id) ON DELETE CASCADE
37 );
38
39 CREATE TABLE Technologies(
40     tech_name varchar(50) NOT NULL,
41     resume_id int NOT NULL,
42     PRIMARY KEY (tech_name, resume_id),
43     FOREIGN KEY (resume_id) REFERENCES Resume (id) ON DELETE CASCADE
44 );
45
```

#	Time	Action	Message
2	20:02:37	CREATE DATABASE AUAsstDB	1 row(s) affected
3	20:02:37	USE AUAsstDB	0 row(s) affected
4	20:02:37	CREATE TABLE Resume (id int NOT NULL AUTO_INCREMENT, pe...	0 row(s) affected
5	20:02:38	CREATE TABLE Project (id int NOT NULL auto_increment, resume...	0 row(s) affected
6	20:02:38	CREATE TABLE WorkDetails(id int NOT NULL auto_increment, re...	0 row(s) affected
7	20:02:38	CREATE TABLE Technologies(tech_name varchar(50) NOT NULL, ...	0 row(s) affected

## 2. INSERTING INTO DB

The screenshot shows a SQL IDE interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including tables like 'project', 'resume', 'technologies', and 'workdetails'. The main editor displays SQL queries for inserting data into the 'Resume' and 'Project' tables. The 'Output' pane at the bottom shows the execution results, including the time, action, and message for each query.

```
46 INSERT INTO Resume VALUES (
47     NULL,
48     "Harshit",
49     "harshit.chhabra09@gmail.com",
50     "+918125799791",
51     "Ashok Nagar, Hyderabad",
52     "Hyderabad",
53     "Dummy objective",
54     "hobbie1,hobbie2,hobbie3",
55     "cert1,cert2,cert3"
56 );
57
58 INSERT INTO Project VALUES (
59     NULL,
60     (SELECT AUTO_INCREMENT
61      FROM information_schema.TABLES
62      WHERE TABLE_SCHEMA = "auasstdb"
63      AND TABLE_NAME = "resume"
64     )-1,
65     "proj1Name",
66     "proj1Desc",
67     "githubLinkDummy",
68     "tech1,tech2,tech3"
69 );
```

#	Time	Action	Message
7	22:20:00	INSERT INTO Resume VALUES ( NULL, "Harshit", "harshit.chhabra..."	1 row(s) affected
8	22:20:00	INSERT INTO Project VALUES ( NULL, (SELECT AUTO_INCREMENT...	1 row(s) affected
9	22:20:00	INSERT INTO workdetails VALUES ( NULL, (SELECT AUTO_INC...	1 row(s) affected

**\*\* BULK ENTRY OF DATA WAS DONE WITH DUMMY DATA (STATIC INPUTS)**

**\*\* QUERY FOR THE SAME CAN BE FOUND IN THE SQL SCRIPT**

### 3.1 Get list of candidates based in 'Hyderabad' which were added to the system in the past 3 months.

```
139
140 • SELECT FROM resume WHERE location = "Hyderabad"
141 AND DATEDIFF(added_date,(SELECT CURDATE()))<=90;
142
```

id	person_name	email	phone	address	location	objective	hobbies	certifications
1	Harshit	harshit.dhhabra09@gmail.com	+918125799791	Ashok Nagar, Hyderabad	Hyderabad	Dummy objective	hobbie1,hobbie2,hobbie3	cert1,cert2,cert3

### 3.2 Get count of candidates in all cities, who know Javascript and Html.

```
142
143 • SELECT COUNT(*) as Count,location as City from resume r WHERE
144 EXISTS (SELECT * from technologies WHERE r.id = resume_id AND tech_name = "HTML")
145 AND
146 EXISTS (SELECT * from technologies WHERE r.id = resume_id AND tech_name = "Javascript") GROUP BY City;
147
```

Count	City
1	Hyderabad
1	loc2
1	loc3

### 3.3 Get list of all candidates sorted by "maximum number of technologies known to him/her"

```
147
148 • SELECT person_name as CandidateName, (SELECT COUNT(*) FROM technologies WHERE r.id = resume_id)
149 as Total_No_Of_Tech_Known FROM resume r ORDER BY Total_No_Of_Tech_Known desc;
```

CandidateName	Total_No_Of_Tech_Known
Harshit	3
name3	3
name5	3
name2	2
name4	2

### 3.4 Create a view that HR can query everyday to get the new candidates added previous day. (Include all necessary fields required by HR)

```
150
151 • CREATE VIEW AddedYesterday AS
152 SELECT person_name, email, phone, added_date FROM resume WHERE added_date = (SELECT DATE_SUB((SELECT CURDATE()), INTERVAL 1 DAY));
153
154 • SELECT * FROM AddedYesterday;
```

person_name	email	phone	added_date
name2	email2	phone2	2020-01-14
name5	email5	phone5	2020-01-14