

# DDL commands

The screenshot shows a MySQL Workbench interface. The top menu bar includes buttons for RUN, datarun, view, AV, R4, 3NF, bcd, RUN1, and HR\_U. Below the menu is a toolbar with icons for file operations, search, and other database management functions. A status bar at the bottom indicates 100% zoom and the time 1:13.

The main area contains the following DDL commands:

```
1 • USE job_recruitment;
2 • SHOW TABLES;
3 • SELECT * FROM USER;
4 • SELECT * FROM HR_USER;
5 • SELECT * FROM CANDIDATE;
6 • SELECT * FROM JOB_POSTING;
7 • SELECT * FROM RESUME;
8 • SELECT * FROM PROCESSED_RESUME;
9 • SELECT * FROM SCREENING_SKILLS;
10 • SELECT * FROM CANDIDATE_APPLIES_FOR_JOB;
11 • SELECT * FROM AI_SCREENING_RESULT;
12 • SELECT * FROM INTERVIEW;
13
```

Below the commands is a "Result Grid" section titled "Tables\_in\_job\_recruitment". It displays a list of tables:

Tables_in_job_recruitment
AI_SCREENING_RESULT
AI_SCREENING_RESULT_3NF
AI_SCREENING_RESULT_RAW
CANDIDATE
CANDIDATE_APPLIES_FOR_JOB
CANDIDATE_INFO
HR_USER
INTERVIEW
JOB
JOB_POSTING
JOB_REQUIRED_SKILL
PROCESSED_RESUME
RESUME
SCREENING_SKILLS
UnnormalizedOrders
USER

Fig 1 Creating database and Creating Tables

1 • USE job\_recruitment;  
 2 • SHOW TABLES;  
 3 • SELECT \* FROM USER;  
 4 • SELECT \* FROM HR\_USER;  
 5 • SELECT \* FROM CANDIDATE;  
 6 • SELECT \* FROM JOB\_POSTING;  
 7 • SELECT \* FROM RESUME;  
 8 • SELECT \* FROM PROCESSED\_RESUME;  
 9 • SELECT \* FROM SCREENING\_SKILLS;  
 10 • SELECT \* FROM CANDIDATE\_APPLIES\_FOR\_JOB;  
 11 • SELECT \* FROM AI\_SCREENING\_RESULT;  
 12 • SELECT \* FROM INTERVIEW;

13

100% 1:13

Result Grid Filter Rows: Search Edit: Export/Import:

User_ID	Name	Email	Phone	User_Type
1	Alice Johnson	alice.hr@example.com	1234567890	HR
2	Bob Smith	bob.hr@example.com	1234567891	HR
3	Charlie Brown	charlie.candidate@example.com	1234567892	Candidate
4	David Lee	david.candidate@example.com	1234567893	Candidate
5	Eva Green	eva.candidate@example.com	1234567894	Candidate
6	Frank White	frank.hr@example.com	1234567895	HR
7	Grace Adams	grace.candidate@example.com	1234567896	Candidate
8	Henry Ford	henry.candidate@example.com	1234567897	Candidate
9	Ivy Clark	ivy.candidate@example.com	1234567898	Candidate
10	Jack Wilson	jack.hr@example.com	1234567899	HR
21	Test Candidate	testc@example.com	1234567890	Candidate
HULL	HULL	HULL	HULL	HULL

Result 1    USER 2    HR\_USER 3    CANDIDATE 4    JOB\_POSTING 5    RESUME 6    PROCESSED\_RESUME 7    SCREENING\_SKILLS 8    CANDIDATE\_APPLIES\_FOR\_JOB 9

1 • USE job\_recruitment;  
 2 • SHOW TABLES;  
 3 • SELECT \* FROM USER;  
 4 • SELECT \* FROM HR\_USER;  
 5 • SELECT \* FROM CANDIDATE;  
 6 • SELECT \* FROM JOB\_POSTING;  
 7 • SELECT \* FROM RESUME;  
 8 • SELECT \* FROM PROCESSED\_RESUME;  
 9 • SELECT \* FROM SCREENING\_SKILLS;  
 10 • SELECT \* FROM CANDIDATE\_APPLIES\_FOR\_JOB;  
 11 • SELECT \* FROM AI\_SCREENING\_RESULT;  
 12 • SELECT \* FROM INTERVIEW;

13

100% 1:13

Result Grid Filter Rows: Search Edit: Export/Import:

User_ID	Role
1	Recruitment Manager
2	HR Specialist
6	HR Coordinator
10	Talent Acquisition Lead
HULL	HULL

Result 1    USER 2    HR\_USER 3    CANDIDATE 4    JOB\_POSTING 5    RESUME 6    PROCESSED\_RESUME 7    SCREENING\_SKILLS 8

Fig 2 Creating USER and HR\_USER and inserting values

```
RUN | datarun | view | AV | R4 | 3NF | bcd | RUN1 | HR_U | Limit to 1000 rows |      
```

1 USE job\_recruitment;  
2 SHOW TABLES;  
3 SELECT \* FROM USER;  
4 SELECT \* FROM HR\_USER;  
5 SELECT \* FROM CANDIDATE;  
6 SELECT \* FROM JOB\_POSTING;  
7 SELECT \* FROM RESUME;  
8 SELECT \* FROM PROCESSED\_RESUME;  
9 SELECT \* FROM SCREENING\_SKILLS;  
10 SELECT \* FROM CANDIDATE\_APPLIES\_FOR\_JOB;  
11 SELECT \* FROM AI\_SCREENING\_RESULT;  
12 SELECT \* FROM INTERVIEW;  
13

100% 1:13

Result Grid Filter Rows: Search Edit: Export/Import:

User_ID	LinkedIn_Profile	Experience_Years	Skills
3	linkedin.com/in/charliebrown	2	HULL
4	linkedin.com/in/davidlee	15	HULL
5	linkedin.com/in/evagreen	3	HULL
7	linkedin.com/in/graceadams	4	HULL
8	linkedin.com/in/henryford	1	HULL
9	linkedin.com/in/ivyclark	6	HULL
21	HULL	HULL	HULL
	HULL	ROLE	HULL

Result 1    USER 2    HR\_USER 3    CANDIDATE 4    JOB\_POSTING 5    RESUME 6    PROCESSED\_RESUME 7    SCREENING\_SKILLS 8

Fig 3 Creating CANDIDATE and JOB\_POSTING and inserting values

1 • USE job\_recruitment;  
 2 • SHOW TABLES;  
 3 • SELECT \* FROM USER;  
 4 • SELECT \* FROM HR\_USER;  
 5 • SELECT \* FROM CANDIDATE;  
 6 • SELECT \* FROM JOB\_POSTING;  
 7 • SELECT \* FROM RESUME;  
 8 • SELECT \* FROM PROCESSED\_RESUME;  
 9 • SELECT \* FROM SCREENING\_SKILLS;  
 10 • SELECT \* FROM CANDIDATE\_APPLIES\_FOR\_JOB;  
 11 • SELECT \* FROM AI\_SCREENING\_RESULT;  
 12 • SELECT \* FROM INTERVIEW;  
 13

100% 41:10 Result Grid Filter Rows: Search Edit: Export/Import:

Resume_ID	Candidate_ID	Resume_File
1	3	charlie_resume.pdf
2	4	david_resume.pdf
3	5	eva_resume.pdf
4	7	grace_resume.pdf
5	8	henry_resume.pdf
HULL	HULL	HULL

Result 1 USER 2 HR\_USER 3 CANDIDATE 4 JOB\_POSTING 5 RESUME 6 PROCESSED\_RESUME 7 SCREENING\_SKILLS 8 CANDIDATE\_APPLIES\_FOR\_JOB 9

1 • USE job\_recruitment;  
 2 • SHOW TABLES;  
 3 • SELECT \* FROM USER;  
 4 • SELECT \* FROM HR\_USER;  
 5 • SELECT \* FROM CANDIDATE;  
 6 • SELECT \* FROM JOB\_POSTING;  
 7 • SELECT \* FROM RESUME;  
 8 • SELECT \* FROM PROCESSED\_RESUME;  
 9 • SELECT \* FROM SCREENING\_SKILLS;  
 10 • SELECT \* FROM CANDIDATE\_APPLIES\_FOR\_JOB;  
 11 • SELECT \* FROM AI\_SCREENING\_RESULT;  
 12 • SELECT \* FROM INTERVIEW;  
 13

100% 41:10 Result Grid Filter Rows: Search Edit: Export/Import:

Resume_ID	Processed_Text
1	Java developer with 2 years experience
2	Data scientist with Python expertise
3	Project Manager with Agile experience
4	Frontend Developer with React skills
5	Backend Developer proficient in Node.js and AWS
HULL	HULL

Result 1 USER 2 HR\_USER 3 CANDIDATE 4 JOB\_POSTING 5 RESUME 6 PROCESSED\_RESUME 7 SCREENING\_SKILLS 8 CANDIDATE\_APPLIES\_FOR\_JOB 9

Fig 4 Creating table RESUME and PROCESSED\_RESUME and inserting the values

RUN | datarun | view | AV | R4 | 3NF | bcd | RUN1 | HR\_U |

```

1 • USE job_recruitment;
2 • SHOW TABLES;
3 • SELECT * FROM USER;
4 • SELECT * FROM HR_USER;
5 • SELECT * FROM CANDIDATE;
6 • SELECT * FROM JOB_POSTING;
7 • SELECT * FROM RESUME;
8 • SELECT * FROM PROCESSED_RESUME;
9 • SELECT * FROM SCREENING_SKILLS;
10 • SELECT * FROM CANDIDATE_APPLIES_FOR_JOB;
11 • SELECT * FROM AI_SCREENING_RESULT;
12 • SELECT * FROM INTERVIEW;
13

```

100% 41:10

Result Grid Filter Rows: Search Edit: Export/Import:

Screening_ID	Resume_ID	Skill_Name
2	1	Java
3	1	SQL
4	2	Python
5	2	Data Science
6	3	Agile
7	3	Project Management
8	4	React
9	4	JavaScript
10	5	Node.js
11	5	AWS
HULL	HULL	HULL

Result 1 | USER 2 | HR\_USER 3 | CANDIDATE 4 | JOB\_POSTING 5 | RESUME 6 | PROCESSED\_RESUME 7 | SCREENING\_SKILLS 8 | CANDIDATE\_APPLIES\_FOR\_JOB 9

RUN | datarun | view | AV | R4 | 3NF | bcd | RUN1 | HR\_U |

```

1 • USE job_recruitment;
2 • SHOW TABLES;
3 • SELECT * FROM USER;
4 • SELECT * FROM HR_USER;
5 • SELECT * FROM CANDIDATE;
6 • SELECT * FROM JOB_POSTING;
7 • SELECT * FROM RESUME;
8 • SELECT * FROM PROCESSED_RESUME;
9 • SELECT * FROM SCREENING_SKILLS;
10 • SELECT * FROM CANDIDATE_APPLIES_FOR_JOB;
11 • SELECT * FROM AI_SCREENING_RESULT;
12 • SELECT * FROM INTERVIEW;
13

```

100% 41:10

Result Grid Filter Rows: Search Edit: Export/Import:

Candidate_ID	Job_ID	Application_Da...	Resume_ID
3	1	2025-04-01	1
4	2	2025-04-01	2
5	3	2025-04-02	3
7	4	2025-04-03	4
8	5	2025-04-04	5
HULL	HULL	HULL	HULL

Result 1 | USER 2 | HR\_USER 3 | CANDIDATE 4 | JOB\_POSTING 5 | RESUME 6 | PROCESSED\_RESUME 7 | SCREENING\_SKILLS 8 | CANDIDATE\_APPLIES\_FOR\_JOB 9

Fig 5 Creating SCREENING\_SKILLS and CANDIDATE\_APPLIES\_FOR\_JOB and inserting values

```
1 ▪ USE job_recruitment;
2 ▪ SHOW TABLES;
3 ▪ SELECT * FROM USER;
4 ▪ SELECT * FROM HR_USER;
5 ▪ SELECT * FROM CANDIDATE;
6 ▪ SELECT * FROM JOB_POSTING;
7 ▪ SELECT * FROM RESUME;
8 ▪ SELECT * FROM PROCESSED_RESUME;
9 ▪ SELECT * FROM SCREENING_SKILLS;
10 ▪ SELECT * FROM CANDIDATE_APPLIES_FOR_JOB;
11 ▪ SELECT * FROM AI_SCREENING_RESULT;
12 ▪ SELECT * FROM INTERVIEW;
13
```

```
1 • USE job_recruitment;
2 • SHOW TABLES;
3 • SELECT * FROM USER;
4 • SELECT * FROM HR_USER;
5 • SELECT * FROM CANDIDATE;
6 • SELECT * FROM JOB_POSTING;
7 • SELECT * FROM RESUME;
8 • SELECT * FROM PROCESSED_RESUME;
9 • SELECT * FROM SCREENING_SKILLS;
10 • SELECT * FROM CANDIDATE_APPLIES_FOR_JOB;
11 • SELECT * FROM AI_SCREENING_RESULT;
12 • SELECT * FROM INTERVIEW;
```

Fig 6 Creating AI SCREENING RESULT and INTERVIEW and inserting values

datarun | view | AV | R4 | 3NF | bcd | RUN1

Limit to 1000 rows

```

1 USE job_recruitment;
2 SHOW COLUMNS FROM USER;
3 SHOW COLUMNS FROM HR_USER;
4 SHOW COLUMNS FROM CANDIDATE;
5 SHOW COLUMNS FROM JOB_POSTING;
6 SHOW COLUMNS FROM RESUME;
7 SHOW COLUMNS FROM PROCESSED_RESUME;
8 SHOW COLUMNS FROM SCREENING_SKILLS;
9 SHOW COLUMNS FROM CANDIDATE_APPLIES_FOR_JOB;
10 SHOW COLUMNS FROM AI_SCREENING_RESULT;
11 SHOW COLUMNS FROM INTERVIEW;
12

```

100% 13:11

**Result Grid** Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
User_ID	int	NO	PRI	HULL	auto_increment
Name	varchar(100)	NO		HULL	
Email	varchar(100)	NO	UNI	HULL	
Phone	varchar(15)	YES		HULL	
User_Type	enum('HR','C...')	NO		HULL	

datarun | view | AV | R4 | 3NF | bcd | RUN1

Limit to 1000 rows

```

1 USE job_recruitment;
2 SHOW COLUMNS FROM USER;
3 SHOW COLUMNS FROM HR_USER;
4 SHOW COLUMNS FROM CANDIDATE;
5 SHOW COLUMNS FROM JOB_POSTING;
6 SHOW COLUMNS FROM RESUME;
7 SHOW COLUMNS FROM PROCESSED_RESUME;
8 SHOW COLUMNS FROM SCREENING_SKILLS;
9 SHOW COLUMNS FROM CANDIDATE_APPLIES_FOR_JOB;
10 SHOW COLUMNS FROM AI_SCREENING_RESULT;
11 SHOW COLUMNS FROM INTERVIEW;
12

```

29:11

**Result Grid** Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
User_ID	int	NO	PRI	HULL	
Role	varchar(50)	YES		HULL	

Result 1 Result 2 Result 3 Result 4 Result 5 Result 6

Fig 7 Displaying USER and HR\_USER table

datarun | view | AV | R4 | 3NF | bcd | RUN1

Limit to 1000 rows

```

1 • USE job_recruitment;
2 • SHOW COLUMNS FROM USER;
3 • SHOW COLUMNS FROM HR_USER;
4 • SHOW COLUMNS FROM CANDIDATE;
5 • SHOW COLUMNS FROM JOB_POSTING;
6 • SHOW COLUMNS FROM RESUME;
7 • SHOW COLUMNS FROM PROCESSED_RESUME;
8 • SHOW COLUMNS FROM SCREENING_SKILLS;
9 • SHOW COLUMNS FROM CANDIDATE_APPLIES_FOR_JOB;
10 • SHOW COLUMNS FROM AI_SCREENING_RESULT;
11 • SHOW COLUMNS FROM INTERVIEW;
12

```

39:10

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
User_ID	int	NO	PRI	NULL	
LinkedIn_Profile	varchar(255)	YES		NULL	
Experience_Years	int	YES		NULL	
Skills	text	YES		NULL	

Result 1 Result 2 Result 3 Result 4 Result 5 Result 6

datarun | view | AV | R4 | 3NF | bcd | RUN1

Limit to 1000 rows

```

1 • USE job_recruitment;
2 • SHOW COLUMNS FROM USER;
3 • SHOW COLUMNS FROM HR_USER;
4 • SHOW COLUMNS FROM CANDIDATE;
5 • SHOW COLUMNS FROM JOB_POSTING;
6 • SHOW COLUMNS FROM RESUME;
7 • SHOW COLUMNS FROM PROCESSED_RESUME;
8 • SHOW COLUMNS FROM SCREENING_SKILLS;
9 • SHOW COLUMNS FROM CANDIDATE_APPLIES_FOR_JOB;
10 • SHOW COLUMNS FROM AI_SCREENING_RESULT;
11 • SHOW COLUMNS FROM INTERVIEW;
12

```

1:12

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
Job_ID	int	NO	PRI	NULL	
HR_ID	int	YES	MUL	NULL	auto_increment
Job_Title	varchar(100)	NO		NULL	
Job_Description	text	YES		NULL	
Required_Skills	text	YES		NULL	
Experience_Level	varchar(50)	YES		NULL	

Result 1 Result 2 Result 3 Result 4 Result 5 Result 6

Fig 8 Displaying CANDIDATE and JOB\_POSTING table

datarun | view | AV | R4 | 3NF | bcd | RUN1

Limit to 1000 rows

```

1 USE job_recruitment;
2 SHOW COLUMNS FROM USER;
3 SHOW COLUMNS FROM HR_USER;
4 SHOW COLUMNS FROM CANDIDATE;
5 SHOW COLUMNS FROM JOB_POSTING;
6 SHOW COLUMNS FROM RESUME;
7 SHOW COLUMNS FROM PROCESSED_RESUME;
8 SHOW COLUMNS FROM SCREENING_SKILLS;
9 SHOW COLUMNS FROM CANDIDATE_APPLIES_FOR_JOB;
10 SHOW COLUMNS FROM AI_SCREENING_RESULT;
11 SHOW COLUMNS FROM INTERVIEW;
12
  
```

Result Grid | Filter Rows: | Search | Export:

Field	Type	Null	Key	Default	Extra
Resume_ID	int	NO	PRI	NULL	auto_increment
Candidate_ID	int	YES	MUL	NULL	
Resume_File	text	YES		NULL	

Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Result 6 | Result 6

datarun | view | AV | R4 | 3NF | bcd | RUN1

Limit to 1000 rows

```

1 USE job_recruitment;
2 SHOW COLUMNS FROM USER;
3 SHOW COLUMNS FROM HR_USER;
4 SHOW COLUMNS FROM CANDIDATE;
5 SHOW COLUMNS FROM JOB_POSTING;
6 SHOW COLUMNS FROM RESUME;
7 SHOW COLUMNS FROM PROCESSED_RESUME;
8 SHOW COLUMNS FROM SCREENING_SKILLS;
9 SHOW COLUMNS FROM CANDIDATE_APPLIES_FOR_JOB;
10 SHOW COLUMNS FROM AI_SCREENING_RESULT;
11 SHOW COLUMNS FROM INTERVIEW;
12
  
```

Result Grid | Filter Rows: | Search | Export:

Field	Type	Null	Key	Default	Extra
Resume_ID	int	NO	PRI	NULL	
Processed_Text	text	YES		NULL	

Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Result 6 | Result 6

Fig 9 Displaying RESUME and PROCESSED\_RESUME table

datarun | view | AV | R4 | 3NF | bcd | RUN1

Limit to 1000 rows

```

1 • USE job_recruitment;
2 • SHOW COLUMNS FROM USER;
3 • SHOW COLUMNS FROM HR_USER;
4 • SHOW COLUMNS FROM CANDIDATE;
5 • SHOW COLUMNS FROM JOB_POSTING;
6 • SHOW COLUMNS FROM RESUME;
7 • SHOW COLUMNS FROM PROCESSED_RESUME;
8 • SHOW COLUMNS FROM SCREENING_SKILLS;
9 • SHOW COLUMNS FROM CANDIDATE_APPLIES_FOR_JOB;
10 • SHOW COLUMNS FROM AI_SCREENING_RESULT;
11 • SHOW COLUMNS FROM INTERVIEW;
12

```

Result Grid | Filter Rows: | Search | Export:

Field	Type	Null	Key	Default	Extra
Screening_ID	int	NO	PRI	NULL	
Resume_ID	int	YES	MUL	NULL	
Skill_Name	varchar(255)	YES		NULL	

Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Result 6

datarun | view | AV | R4 | 3NF | bcd | RUN1

Limit to 1000 rows

```

1 • USE job_recruitment;
2 • SHOW COLUMNS FROM USER;
3 • SHOW COLUMNS FROM HR_USER;
4 • SHOW COLUMNS FROM CANDIDATE;
5 • SHOW COLUMNS FROM JOB_POSTING;
6 • SHOW COLUMNS FROM RESUME;
7 • SHOW COLUMNS FROM PROCESSED_RESUME;
8 • SHOW COLUMNS FROM SCREENING_SKILLS;
9 • SHOW COLUMNS FROM CANDIDATE_APPLIES_FOR_JOB;
10 • SHOW COLUMNS FROM AI_SCREENING_RESULT;
11 • SHOW COLUMNS FROM INTERVIEW;
12

```

Result Grid | Filter Rows: | Search | Export:

Field	Type	Null	Key	Default	Extra
Candidate_ID	int	NO	PRI	NULL	
Job_ID	int	NO	PRI	NULL	
Application_Date	date	YES		NULL	
Resume_ID	int	YES	MUL	NULL	

Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Result 6

Fig 10 Displaying SCREENING\_SKILLS and CANDIDATE\_APPLIES\_FOR\_JOB table

1 USE job\_recruitment;  
 2 SHOW COLUMNS FROM USER;  
 3 SHOW COLUMNS FROM HR\_USER;  
 4 SHOW COLUMNS FROM CANDIDATE;  
 5 SHOW COLUMNS FROM JOB\_POSTING;  
 6 SHOW COLUMNS FROM RESUME;  
 7 SHOW COLUMNS FROM PROCESSED\_RESUME;  
 8 SHOW COLUMNS FROM SCREENING\_SKILLS;  
 9 SHOW COLUMNS FROM CANDIDATE\_APPLIES\_FOR\_JOB;  
 10 SHOW COLUMNS FROM AI\_SCREENING\_RESULT;  
 11 SHOW COLUMNS FROM INTERVIEW;  
 12

Result 2 | Result 3 | Result 4 | Result 5 | Result 6 | Result 7

1 USE job\_recruitment;  
 2 SHOW COLUMNS FROM USER;  
 3 SHOW COLUMNS FROM HR\_USER;  
 4 SHOW COLUMNS FROM CANDIDATE;  
 5 SHOW COLUMNS FROM JOB\_POSTING;  
 6 SHOW COLUMNS FROM RESUME;  
 7 SHOW COLUMNS FROM PROCESSED\_RESUME;  
 8 SHOW COLUMNS FROM SCREENING\_SKILLS;  
 9 SHOW COLUMNS FROM CANDIDATE\_APPLIES\_FOR\_JOB;  
 10 SHOW COLUMNS FROM AI\_SCREENING\_RESULT;  
 11 SHOW COLUMNS FROM INTERVIEW;  
 12

Result 3 | Result 4 | Result 5 | Result 6 | Result 7 | Result 8

Fig 11 Displaying AI\_SCREENING\_RESULT and INTERVIEW table

# DML commands

The screenshot shows a MySQL Workbench interface with a query editor and a result grid. The query editor contains the following DML command:

```
1 use job_recruitment;
2 INSERT INTO USER (Name, Email, Phone, User_Type)
3 VALUES ('Joe', 'joe.candidate@example.com', '1234500000', 'Candidate');
4 select * from user;
```

The result grid displays the contents of the USER table, including the newly inserted record for 'Joe'.

User_ID	Name	Email	Phone	User_Type
1	Alice Johnson	alice.hr@example.com	1234567890	HR
2	Bob Smith	bob.hr@example.com	1234567891	HR
3	Charlie Brown	charlie.candidate@example.com	1234567892	Candidate
4	David Lee	david.candidate@example.com	1234567893	Candidate
5	Eva Green	eva.candidate@example.com	1234567894	Candidate
6	Frank White	frank.hr@example.com	1234567895	HR
7	Grace Adams	grace.candidate@example.com	1234567896	Candidate
8	Henry Ford	henry.candidate@example.com	1234567897	Candidate
9	Ivy Clark	ivy.candidate@example.com	1234567898	Candidate
10	Jack Wilson	jack.hr@example.com	1234567899	HR
21	Test Candidate	testc@example.com	1234567890	Candidate
22	Joe	joe.candidate@example.com	1234500000	Candidate
HULL	HULL	HULL	HULL	HULL

Fig 1 Inserting a new record into the table

The screenshot shows a MySQL Workbench interface with a query editor and a result grid. The query editor contains the following DML command:

```
1 use job_recruitment;
2 UPDATE USER
3 SET Email = 'updated.email@example.com',
4     Phone = '9999998888'
5 WHERE User_ID = 3;
6
7 SELECT * FROM USER WHERE User_ID = 3;
8
```

The result grid displays the contents of the USER table, showing the updated record for User ID 3.

User_ID	Name	Email	Phone	User_Type
1	Alice Johnson	alice.hr@example.com	1234567890	HR
2	Bob Smith	bob.hr@example.com	1234567891	HR
3	Charlie Brown	updated.email@example.com	9999998888	Candidate
4	David Lee	david.candidate@example.com	1234567893	Candidate
5	Eva Green	eva.candidate@example.com	1234567894	Candidate
6	Frank White	frank.hr@example.com	1234567895	HR
7	Grace Adams	grace.candidate@example.com	1234567896	Candidate
8	Henry Ford	henry.candidate@example.com	1234567897	Candidate
9	Ivy Clark	ivy.candidate@example.com	1234567898	Candidate
10	Jack Wilson	jack.hr@example.com	1234567899	HR
21	Test Candidate	testc@example.com	1234567890	Candidate
22	Joe	joe.candidate@example.com	1234500000	Candidate
HULL	HULL	HULL	HULL	HULL

Fig 2 Updating a new record into the table

The screenshot shows a MySQL Workbench session titled 'datarun'. The SQL editor contains the following code:

```

1 • use job_recruitment;
2 • DELETE FROM SCREENING_SKILLS
3 WHERE Resume_ID IN (SELECT Resume_ID FROM PROCESSED_RESUME WHERE Resume_ID IN (SELECT Resume_ID FROM RESUME WHERE Candidate_ID = 3));
4 • DELETE FROM PROCESSED_RESUME
5 WHERE Resume_ID IN (SELECT Resume_ID FROM RESUME WHERE Candidate_ID = 3);
6 • DELETE FROM AI_SCREENNING_RESULT
7 WHERE Candidate_ID = 3;
8 • DELETE FROM CANDIDATE_APPLIES_FOR_JOB
9 WHERE Candidate_ID = 3;
10 • DELETE FROM RESUME
11 WHERE Candidate_ID = 3;
12 • DELETE FROM CANDIDATE
13 WHERE User_ID = 3;
14 • DELETE FROM USER
15 WHERE User_ID = 3;
16 • SELECT * FROM USER ;
17

```

The Result Grid displays the following data:

User_ID	Name	Email	Phone	User_Type
1	Alice Johnson	alice.hr@example.com	1234567890	HR
2	Bob Smith	bob.hr@example.com	1234567891	HR
4	David Lee	david.candidate@example.com	1234567893	Candidate
5	Eva Green	eva.candidate@example.com	1234567894	Candidate
6	Frank White	frank.hr@example.com	1234567895	HR
7	Grace Adams	grace.candidate@example.com	1234567896	Candidate
8	Henry Ford	henry.candidate@example.com	1234567897	Candidate
9	Ivy Clark	ivy.candidate@example.com	1234567898	Candidate
10	Jack Wilson	jack.hr@example.com	1234567899	HR
21	Test Candidate	testc@example.com	1234567890	Candidate
22	Joe	joe.candidate@example.com	1234500000	Candidate
HULL	HULL	HULL	HULL	HULL

Fig 3 Deleting a record into the table

The screenshot shows a MySQL Workbench session titled 'datarun'. The SQL editor contains the following code:

```

1 • use job_recruitment;
2
3 • SELECT * FROM USER ;
4

```

The Result Grid displays the same data as Fig 3:

User_ID	Name	Email	Phone	User_Type
1	Alice Johnson	alice.hr@example.com	1234567890	HR
2	Bob Smith	bob.hr@example.com	1234567891	HR
4	David Lee	david.candidate@example.com	1234567893	Candidate
5	Eva Green	eva.candidate@example.com	1234567894	Candidate
6	Frank White	frank.hr@example.com	1234567895	HR
7	Grace Adams	grace.candidate@example.com	1234567896	Candidate
8	Henry Ford	henry.candidate@example.com	1234567897	Candidate
9	Ivy Clark	ivy.candidate@example.com	1234567898	Candidate
10	Jack Wilson	jack.hr@example.com	1234567899	HR
21	Test Candidate	testc@example.com	1234567890	Candidate
22	Joe	joe.candidate@example.com	1234500000	Candidate
HULL	HULL	HULL	HULL	HULL

Fig 4 Selecting and viewing data from the table

datarun view AV 3NF bcd RUN1 A

```

1 •   SELECT
2     u.User_ID,
3     u.Name,
4     c.LinkedIn_Profile,
5     r.Resume_File
6   FROM USER u
7   JOIN CANDIDATE c ON u.User_ID = c.User_ID
8   LEFT JOIN RESUME r ON c.User_ID = r.Candidate_ID;
9

```

Result Grid Filter Rows: Search Export:

User_ID	Name	LinkedIn_Profile	Resume_File
3	Charlie Brown	NULL	NULL
4	David Lee	<a href="https://linkedin.com/in/davidlee">linkedin.com/in/davidlee</a>	<a href="#">david_resume.pdf</a>
5	Eva Green	<a href="https://linkedin.com/in/evagreen">linkedin.com/in/evagreen</a>	<a href="#">eva_resume.pdf</a>
7	Grace Adams	<a href="https://linkedin.com/in/graceadams">linkedin.com/in/graceadams</a>	<a href="#">grace_resume.pdf</a>
8	Henry Ford	<a href="https://linkedin.com/in/henryford">linkedin.com/in/henryford</a>	<a href="#">henry_resume.pdf</a>
9	Ivy Clark	<a href="https://linkedin.com/in/ivyclark">linkedin.com/in/ivyclark</a>	NULL
21	Test Candidate	NULL	NULL
22	Joe	NULL	NULL

Fig 5 Left join

datarun view AV 3NF bcd RUN1 AV view

```

1 •   SELECT
2     u.User_ID,
3     u.Name,
4     c.LinkedIn_Profile,
5     r.Resume_File
6   FROM RESUME r
7   RIGHT JOIN CANDIDATE c ON r.Candidate_ID = c.User_ID
8   RIGHT JOIN USER u ON c.User_ID = u.User_ID;
9

```

Result Grid Filter Rows: Search Export:

User_ID	Name	LinkedIn_Profile	Resume_File
1	Alice Johnson	NULL	NULL
2	Bob Smith	NULL	NULL
3	Charlie Brown	NULL	NULL
4	David Lee	<a href="https://linkedin.com/in/davidlee">linkedin.com/in/davidlee</a>	<a href="#">david_resume.pdf</a>
5	Eva Green	<a href="https://linkedin.com/in/evagreen">linkedin.com/in/evagreen</a>	<a href="#">eva_resume.pdf</a>
6	Frank White	NULL	NULL
7	Grace Adams	<a href="https://linkedin.com/in/graceadams">linkedin.com/in/graceadams</a>	<a href="#">grace_resume.pdf</a>
8	Henry Ford	<a href="https://linkedin.com/in/henryford">linkedin.com/in/henryford</a>	<a href="#">henry_resume.pdf</a>
9	Ivy Clark	<a href="https://linkedin.com/in/ivyclark">linkedin.com/in/ivyclark</a>	NULL
10	Jack Wilson	NULL	NULL
21	Test Candidate	NULL	NULL
22	Joe	NULL	NULL

Fig 6 Right join

The screenshot shows a database interface with a query editor at the top and a result grid below. The query is:

```

1  SELECT
2      u.User_ID,
3      u.Name,
4      c.LinkedIn_Profile,
5      r.Resume_File
6  FROM USER u
7  INNER JOIN CANDIDATE c ON u.User_ID = c.User_ID
8  INNER JOIN RESUME r ON c.User_ID = r.Candidate_ID;
9

```

The result grid displays four columns: User\_ID, Name, LinkedIn\_Profile, and Resume\_File. The data is as follows:

User_ID	Name	LinkedIn_Profile	Resume_File
4	David Lee	linkedin.com/in/davidlee	david_resume.pdf
5	Eva Green	linkedin.com/in/evagreen	eva_resume.pdf
7	Grace Adams	linkedin.com/in/graceadams	grace_resume.pdf
8	Henry Ford	linkedin.com/in/henryford	henry_resume.pdf

Fig 7 Inner join

The screenshot shows a database interface with a query editor at the top and a result grid below. The query is:

```

4      u.Name,
5      c.LinkedIn_Profile,
6      r.Resume_File
7  FROM USER u
8  LEFT JOIN CANDIDATE c ON u.User_ID = c.User_ID
9  LEFT JOIN RESUME r ON c.User_ID = r.Candidate_ID
10
11 UNION
12
13 SELECT
14     u.User_ID,
15     u.Name,
16     c.LinkedIn_Profile,
17     r.Resume_File
18  FROM USER u
19  RIGHT JOIN CANDIDATE c ON u.User_ID = c.User_ID
20  RIGHT JOIN RESUME r ON c.User_ID = r.Candidate_ID;
21

```

The result grid displays four columns: User\_ID, Name, LinkedIn\_Profile, and Resume\_File. The data is as follows:

User_ID	Name	LinkedIn_Profile	Resume_File
1	Alice Johnson	NULL	NULL
2	Bob Smith	NULL	NULL
3	Charlie Brown	NULL	NULL
4	David Lee	linkedin.com/in/davidlee	david_resume.pdf
5	Eva Green	linkedin.com/in/evagreen	eva_resume.pdf
6	Frank White	NULL	NULL
7	Grace Adams	linkedin.com/in/graceadams	grace_resume.pdf
8	Henry Ford	linkedin.com/in/henryford	henry_resume.pdf
9	Ivy Clark	linkedin.com/in/ivyclark	NULL
10	Jack Wilson	NULL	NULL
21	Test Candidate	NULL	NULL
22	Joe	NULL	NULL

Fig 8 Outer join

# COMPLEX QUERIES

The screenshot shows the MySQL Workbench interface with a query editor and history panel. The query editor contains the following SQL code:

```
1 DELIMITER $$  
2  
3 DROP TRIGGER IF EXISTS trg_after_user_insert $$  
4  
5 CREATE TRIGGER trg_after_user_insert  
6 AFTER INSERT ON USER  
7 FOR EACH ROW  
8 BEGIN  
9     IF NEW.User_Type = 'Candidate' THEN  
10         INSERT INTO CANDIDATE (User_ID) VALUES (NEW.User_ID);  
11     ELSEIF NEW.User_Type = 'HR' THEN  
12         INSERT INTO HR_USER (User_ID, Role) VALUES (NEW.User_ID, 'Recruiter');  
13     END IF;  
14 END $$  
15  
16 DELIMITER ;  
17
```

The history panel shows a single entry:

Date	Time	Statement
2025-05-04	23:56:14	CREATE TRIGGER trg_after_user_insert

Fig 1 Trigger implementation

The screenshot shows the MySQL Workbench interface with a query editor and result grid. The query editor contains the following SQL code:

```
1 DELIMITER $$  
2 • CREATE PROCEDURE GetShortlistedCandidates()  
3 BEGIN  
4     SELECT * FROM shortlisted_candidates;  
5 END $$  
6 DELIMITER ;  
7 • CALL GetShortlistedCandidates();  
8
```

The result grid displays the output of the stored procedure:

Name	Resume_File	Candidate_ID
David Lee	david_resume.pdf	4
Grace Adams	grace_resume.pdf	7

Fig 2 Stored-Procedure implementation

The screenshot shows the MySQL Workbench interface with a code editor and a result grid.

```

1
2     DELIMITER //
3
4     CREATE PROCEDURE MarkAllResumes()
5     BEGIN
6         DECLARE done INT DEFAULT FALSE;
7         DECLARE rid INT;
8         DECLARE cur CURSOR FOR SELECT Resume_ID FROM Resume;
9         DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
10
11        OPEN cur;
12
13        read_loop: LOOP
14            FETCH cur INTO rid;
15            IF done THEN
16                LEAVE read_loop;
17            END IF;
18            INSERT INTO Resume_Status (Resume_ID, Status) VALUES (rid, 'Processed');
19        END LOOP;
20
21        CLOSE cur;
22    END;
23    //
24
25    DELIMITER ;
26    SELECT * FROM Resume_Status;
27

```

Result Grid:

Resume_ID	Status
2	Processed
3	Processed
4	Processed
5	Processed
NULL	NULL

Fig 3 Cursor implementation

The screenshot shows the MySQL Workbench interface with a code editor and a result grid.

```

1     CREATE VIEW shortlisted_candidates AS
2         SELECT u.Name, r.Resume_File, r.Candidate_ID
3         FROM User u
4         JOIN Resume r ON u.User_ID = r.Candidate_ID
5         JOIN AI_SCREENING_RESULT as sr ON r.Candidate_ID = sr.Candidate_ID
6         WHERE sr.Status = 'Shortlisted';
7
8     SELECT * FROM shortlisted_candidates;

```

Result Grid:

Name	Resume_File	Candidate_ID
David Lee	david_resume.pdf	4
Grace Adams	grace_resume.pdf	7

Fig 4 View implementation

# NORMALIZATIONS

```

1 • CREATE TABLE Candidate_Job_UNF (
2     Candidate_Name VARCHAR(100),
3     Phone VARCHAR(15),
4     LinkedIn_Profile TEXT,
5     Job_Title TEXT,
6     Relevance_Percentage FLOAT,
7     Score FLOAT,
8     Status ENUM('Shortlisted', 'Rejected'),
9     Remark TEXT
10 );
11 • INSERT INTO Candidate_Job_UNF VALUES
12     ('Alice Sharma', '9876543210', 'linkedin.com/in/alice', 'Data Scientist', 85.5, 8.5, 'Shortlisted', 'Strong ML experience'),
13     ('Bob Verma', '9876543211', 'linkedin.com/in/bob', 'Backend Developer', 78.0, 7.8, 'Shortlisted', 'Good backend skills'),
14     ('Charlie Yadav', '9876543212', 'linkedin.com/in/charlie', 'Frontend Developer', 60.0, 6.0, 'Rejected', 'Weak frontend knowledge');
15 • SELECT * FROM Candidate_Job_UNF;
16

```

Result Grid

Candidate_Name	Phone	LinkedIn_Profile	Job_Title	Relevance_Percentage	Score	Status	Remark
Alice Sharma	9876543210	linkedin.com/in/alice	Data Scientist	85.5	8.5	Shortlisted	Strong ML experience
Bob Verma	9876543211	linkedin.com/in/bob	Backend Developer	78	7.8	Shortlisted	Good backend skills
Charlie Yadav	9876543212	linkedin.com/in/charlie	Frontend Developer	60	6	Rejected	Weak frontend knowledge
Alice Sharma	9876543210	linkedin.com/in/alice	Data Scientist	85.5	8.5	Shortlisted	Strong ML experience
Bob Verma	9876543211	linkedin.com/in/bob	Backend Developer	78	7.8	Shortlisted	Good backend skills
Charlie Yadav	9876543212	linkedin.com/in/charlie	Frontend Developer	60	6	Rejected	Weak frontend knowledge

Fig 1 Unnormalized Table

```

1 • CREATE TABLE AI_SCREENING_RESULT_INF (
2     Result_ID INT AUTO_INCREMENT PRIMARY KEY,
3     Candidate_ID INT,
4     Job_ID INT,
5     Relevance_Percentage FLOAT,
6     Score FLOAT,
7     Status ENUM('Shortlisted', 'Rejected'),
8     Remark TEXT,
9     FOREIGN KEY (Candidate_ID) REFERENCES CANDIDATE_INFO(Candidate_ID),
10    FOREIGN KEY (Job_ID) REFERENCES JOB(Job_ID)
11
12 • INSERT INTO CANDIDATE_INFO (Name, Phone, LinkedIn_Profile) VALUES
13     ('Alice Sharma', '9876543210', 'linkedin.com/in/alice'),
14     ('Bob Verma', '9876543211', 'linkedin.com/in/bob'),
15     ('Charlie Yadav', '9876543212', 'linkedin.com/in/charlie');
16
17 • INSERT INTO JOB (Job_Title) VALUES
18     ('Data Scientist'),
19     ('Backend Developer'),
20     ('Frontend Developer');
21
22 • INSERT INTO AI_SCREENING_RESULT_INF (Candidate_ID, Job_ID, Relevance_Percentage, Score, Status, Remark) VALUES
23     (1, 1, 85.5, 8.5, 'Shortlisted', 'Strong ML experience'),
24     (2, 2, 78.0, 7.8, 'Shortlisted', 'Good backend skills'),
25     (3, 3, 60.0, 6.0, 'Rejected', 'Weak frontend knowledge');
26
27 • SELECT
28     c.Name, c.Phone, c.LinkedIn_Profile,
29     j.Job_Title, r.Relevance_Percentage, r.Score, r.Status, r.Remark
30     FROM AI_SCREENING_RESULT_INF r
31     JOIN CANDIDATE_INFO c ON r.Candidate_ID = c.Candidate_ID
32     JOIN JOB j ON r.Job_ID = j.Job_ID;
33
34

```

Result Grid

Name	Phone	LinkedIn_Profile	Job_Title	Relevance_Percentage	Score	Status	Remark
Alice Sharma	9876543210	linkedin.com/in/alice	Data Scientist	85.5	8.5	Shortlisted	Strong ML experience
Bob Verma	9876543211	linkedin.com/in/bob	Backend Developer	78	7.8	Shortlisted	Good backend skills
Charlie Yadav	9876543212	linkedin.com/in/charlie	Frontend Developer	60	6	Rejected	Weak frontend knowledge

Fig 2 1NF Table

```
AV 3NF bcd 2NF* TNF UNF
Limit to 1000 rows
CREATE TABLE IF NOT EXISTS AT_SCREENING_RESULT_2NF (
    Result_ID INT AUTO_INCREMENT PRIMARY KEY,
    Candidate_ID INT,
    Job_ID INT,
    Relevance_Percentage FLOAT,
    Score FLOAT,
    Status ENUM('Shortlisted', 'Rejected'),
    Remark TEXT,
    FOREIGN KEY (Candidate_ID) REFERENCES CANDIDATE_INFO(Candidate_ID),
    FOREIGN KEY (Job_ID) REFERENCES JOB(Job_ID)
);

INSERT INTO CANDIDATE_INFO (Name, Phone, LinkedIn_Profile) VALUES
    ('Alice Sharma', '9876543210', 'linkedin.com/in/alice'),
    ('Bob Verma', '9876543211', 'linkedin.com/in/bob'),
    ('Charlie Yadav', '9876543212', 'linkedin.com/in/charlie');

INSERT INTO JOB (Job_Title) VALUES
    ('Data Scientist'),
    ('Backend Developer'),
    ('Frontend Developer');

INSERT INTO AT_SCREENING_RESULT_2NF (
    Candidate_ID, Job_ID, Relevance_Percentage, Score, Status, Remark
) VALUES
    (1, 1, 85.5, 8.5, 'Shortlisted', 'Strong ML experience'),
    (2, 2, 78.0, 7.8, 'Shortlisted', 'Good backend skills'),
    (3, 3, 60.0, 6.0, 'Rejected', 'Weak frontend knowledge');

SELECT
    c.Name, c.Phone, c.LinkedIn_Profile,
    j.Job_Title,
    r.Relevance_Percentage, r.Score, r.Status, r.Remark
FROM AT_SCREENING_RESULT_2NF r
JOIN CANDIDATE_INFO c ON r.Candidate_ID = c.Candidate_ID
JOIN JOB j ON r.Job_ID = j.Job_ID;

```

Fig 3 2NF Table

Fig 4 3NF Table

# CONCURRENCY CONTROL

The screenshot shows a MySQL Workbench interface with a query editor and results grid. The query editor contains the following SQL code:

```
1 • use job_recruitment;
2 • START TRANSACTION;
3
4 • INSERT INTO CANDIDATE_INFO (Name, Phone, LinkedIn_Profile)
5   VALUES ('David Kumar', '9876543213', 'linkedin.com/in/david');
6
7 • INSERT INTO JOB (Job_Title)
8   VALUES ('DevOps Engineer');
9
10 • INSERT INTO AI_SCREENING_RESULT_2NF (
11     Candidate_ID, Job_ID, Relevance_Percentage, Score, Status, Remark
12   )
13   VALUES (4, 4, 88.0, 8.8, 'Shortlisted', 'Strong CI/CD Knowledge');
14 • COMMIT;
15
16 • SELECT ROW_COUNT();
17
```

The results grid shows the output of the last query:

ROW_COUNT()
1

Fig 1 Transaction Handling

The screenshot shows a MySQL Workbench interface with a query editor and results grid. The query editor contains the following SQL code:

```
1 • START TRANSACTION;
2
3 -- Lock the rows being worked on
4 • SELECT * FROM CANDIDATE_INFO WHERE Candidate_ID = 1 FOR UPDATE;
5 • SELECT * FROM JOB WHERE Job_ID = 1 FOR UPDATE;
6
7 -- Perform safe update
8 • UPDATE AI_SCREENING_RESULT_2NF
9   SET Score = 9.2, Remark = 'Updated score after re-evaluation'
10  WHERE Candidate_ID = 1 AND Job_ID = 1;
11
12 • COMMIT;
13
```

The results grid shows the output of the last query:

Job_ID	Job_Title
1	Data Scientist
NULL	NULL

Fig 2 Concurrency Control

The screenshot shows the MySQL Workbench interface with a SQL editor window. The code in the editor is as follows:

```
1 • use job_recruitment;
2 DELIMITER //
3
4 • CREATE PROCEDURE SafeInsert()
5 BEGIN
6     DECLARE deadlock_occurred INT DEFAULT 0;
7
8     -- Deadlock handler
9     DECLARE CONTINUE HANDLER FOR SQLSTATE '40001'
10        SET deadlock_occurred = 1;
11
12     START TRANSACTION;
13
14     -- Attempt insert (or any operation that may cause deadlock)
15     INSERT INTO AI_SCREENING_RESULT_2NF (
16         Candidate_ID, Job_ID, Relevance_Percentage, Score, Status, Remark
17     ) VALUES (2, 2, 82.5, 8.2, 'Shortlisted', 'Efficient Java performance');
18
19     -- Rollback if deadlock, otherwise commit
20     IF deadlock_occurred = 1 THEN
21         ROLLBACK;
22     ELSE
23         COMMIT;
24     END IF;
25 END //
26
27 DELIMITER ;
28
29 • CALL SafeInsert(); -- Calling the procedure you created for deadlock handling
30
```

The status bar at the bottom indicates "1:28 PM 100% 1000000 STATUS". Below the editor is a "Result Grid" pane with the following table:

Type	Name	Status
InnoDB		====... Result 3

Fig 3 Deadlock Handling