

1. Identify all Space Shuttle missions by the Shuttle Challenger. Display the mission start date, end date, launch site, landing site and mission objectives.

Limit to 50000 rows

```

1 • SELECT mission.missionname AS missionName, spaceship.name AS spaceshipName,
2 mission.startDate, mission.endDate, mission.launchSight, mission.landingSight, objectives.objective FROM
3 objectives, mission, spaceship
4 WHERE objectives.missionId = mission.Id
5 AND mission.spaceshipId = spaceship.id
6 AND spaceship.name LIKE '%Challenger%';
7

```

100% 40:6

Result Grid Filter Rows: Search Export:

	missionName	spaceshipName	startDate	endDate	launchSight	landingSight	objective
▶	STS-6	Space Shuttle Challenger	1983-04-09	1983-04-12	Kennedy Space Center	Pluto	TDRS-A
▶	STS-6	Space Shuttle Challenger	1983-04-09	1983-04-12	Kennedy Space Center	Pluto	Deployment
▶	STS-6	Space Shuttle Challenger	1983-04-09	1983-04-12	Kennedy Space Center	Pluto	Satellites
▶	STS-7	Space Shuttle Challenger	1983-06-18	1983-06-20	Kennedy Space Center	Venus	Communications
▶	STS-7	Space Shuttle Challenger	1983-06-18	1983-06-20	Kennedy Space Center	Venus	Satellites
▶	STS-7	Space Shuttle Challenger	1983-06-18	1983-06-20	Kennedy Space Center	Venus	Deployment
▶	STS-7	Space Shuttle Challenger	1983-06-18	1983-06-20	Kennedy Space Center	Venus	Microgravity
▶	STS-7	Space Shuttle Challenger	1983-06-18	1983-06-20	Kennedy Space Center	Venus	Research
▶	STS-8	Space Shuttle Challenger	1983-08-30	1983-05-09	Kennedy Space Center	Milky Way Galaxy	Communications
▶	STS-8	Space Shuttle Challenger	1983-08-30	1983-05-09	Kennedy Space Center	Milky Way Galaxy	Satellites
▶	STS-8	Space Shuttle Challenger	1983-08-30	1983-05-09	Kennedy Space Center	Milky Way Galaxy	Deployment

2. Identify the missions by astronaut. Display three columns: Astronaut name, number of missions and total mileage. Display one row for each astronaut. Display names in alphabetical order.

```

1 • Select astronauts.firstName, astronauts.lastName, count(mission.id) AS total_missions,
2 sum(mission.mileage) AS total_mileage FROM mission, astronauts, missions_astro
3 WHERE missions_astro.missionId = mission.id AND missions_astro.astronautId = astronauts.id
4 GROUP BY astronauts.id
5 ORDER BY astronauts.firstName ASC;

```

100% 46:3

Result Grid Filter Rows: Search Export:

	firstName	lastName	total_missions	total_mileage
▶	Bob	Dylan	3	1661
▶	Courtney	Dill	1	501
▶	Jeff	Hardy	2	1007
▶	John	Smith	1	542
▶	Koby	White	2	779
▶	Malissa	Shiffman	1	742
▶	Mark	Beltron	2	816
▶	Presh	Takur	1	387
▶	William	lin	1	208
▶	Zack	Hermin	3	1224

3. Identify all missions to Pluto. Display the spaceship name, start and end date of mission and objectives. Order the output by date.

Limit to 50000 rows

```

1 • SELECT spaceship.name, mission.landingSight, mission.startDate,
2 mission.endDate, objectives.objective
3 FROM spaceship, objectives, mission WHERE
4 mission.id = objectives.missionId AND
5 spaceship.id = mission.spaceshipId AND
6 mission.landingSight = 'Pluto'
7 ORDER BY endDate DESC;

```

100% 23:7

Result Grid Filter Rows: Search Export:

	name	landingSight	startDate	endDate	objective
	Space Shuttle Challenger	Pluto	1983-04-09	1983-04-12	TDRS-A
	Space Shuttle Challenger	Pluto	1983-04-09	1983-04-12	Deployment
	Space Shuttle Challenger	Pluto	1983-04-09	1983-04-12	Satellites
	Space Shuttle Columbia	Pluto	1982-11-11	1982-11-16	Communications
	Space Shuttle Columbia	Pluto	1982-11-11	1982-11-16	Satellites
	Space Shuttle Columbia	Pluto	1982-11-11	1982-11-16	Deployment
	Space Shuttle Columbia	Pluto	1982-06-27	1982-07-04	Test flight
	Space Shuttle Columbia	Pluto	1982-03-28	1982-03-30	Test flight

4. Identify the missions by launch site since 1984. Display four columns: Launch site name, site country, site state and number of missions. Display one row for each launch site. (I did 1984 because my dates were older then 2000)

Limit to 50000 rows

```

1 SELECT mission.launchSight, mission.landingSight, mission.sightCountry, mission.sightState,
2 count(mission.launchSight) as num_of_missions FROM
3 mission WHERE mission.startDate > 19840000
4 GROUP BY mission.launchSight, mission.landingSight, mission.sightCountry, mission.sightState;

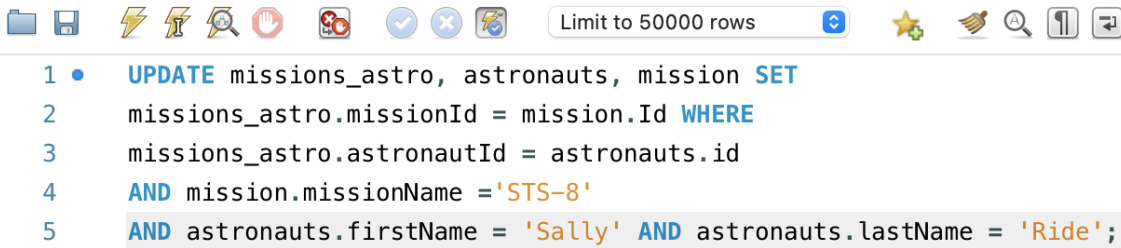
```

100% 94:4

Result Grid Filter Rows: Search Export:

	launchSight	landingSight	sightCount...	sightState	num_of_missio...
►	Kennedy Space Center	Moon	USA	Florida	2

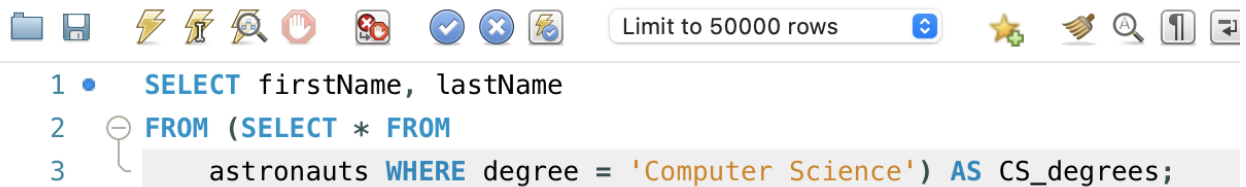
5. Re-assign astronaut Sally Ride from Space Shuttle mission STS-7 to STS-8. Identify the SQL required to implement.



Limit to 50000 rows

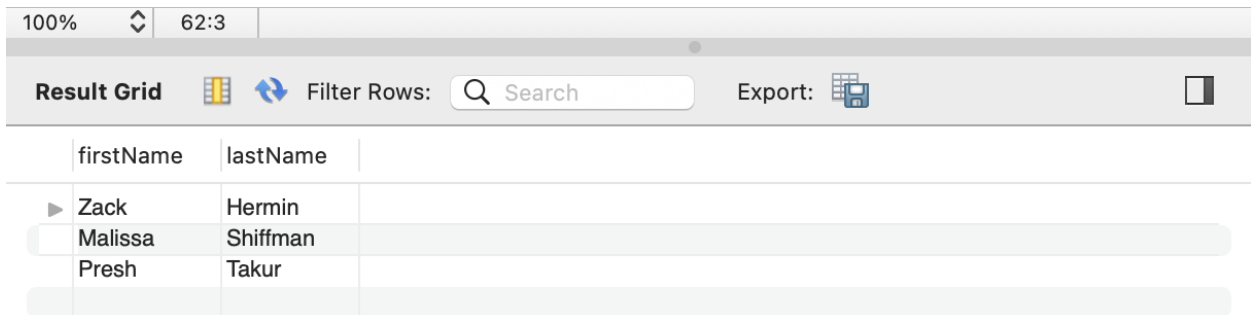
```
1 • UPDATE missions_astro, astronauts, mission SET
2   missions_astro.missionId = mission.Id WHERE
3   missions_astro.astronautId = astronauts.id
4   AND mission.missionName = 'STS-8'
5   AND astronauts.firstName = 'Sally' AND astronauts.lastName = 'Ride';
```

6. Identify astronauts who have a Computer Science degree. Use a nested select to answer this question.



Limit to 50000 rows

```
1 • SELECT firstName, lastName
2   FROM (SELECT * FROM
3         astronauts WHERE degree = 'Computer Science') AS CS_degrees;
```

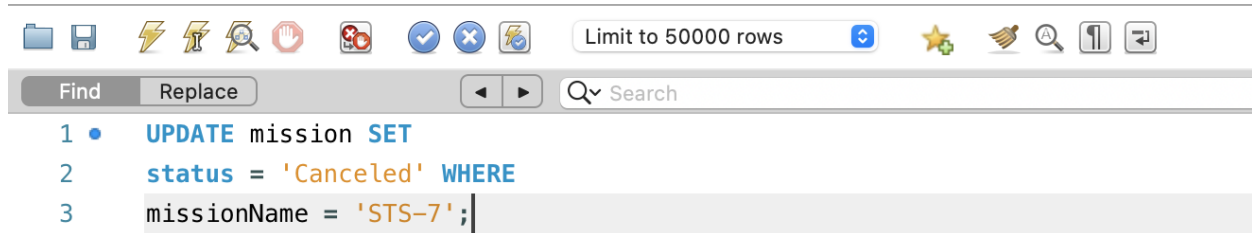


100% 62:3

Result Grid Filter Rows: Search Export:

	firstName	lastName
▶	Zack	Hermin
▶	Malissa	Shiffman
▶	Presh	Takur

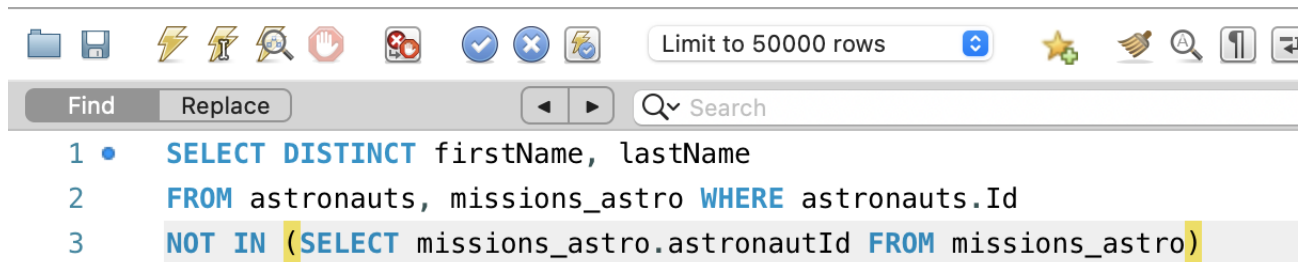
7. Cancel Space Shuttle mission STS-7. Identify the SQL to implement.



The screenshot shows a SQL editor with a toolbar at the top containing icons for file operations, execution, and search. A search bar on the right indicates 'Limit to 50000 rows'. Below the toolbar is a 'Find' and 'Replace' section. The main area contains the following SQL query:

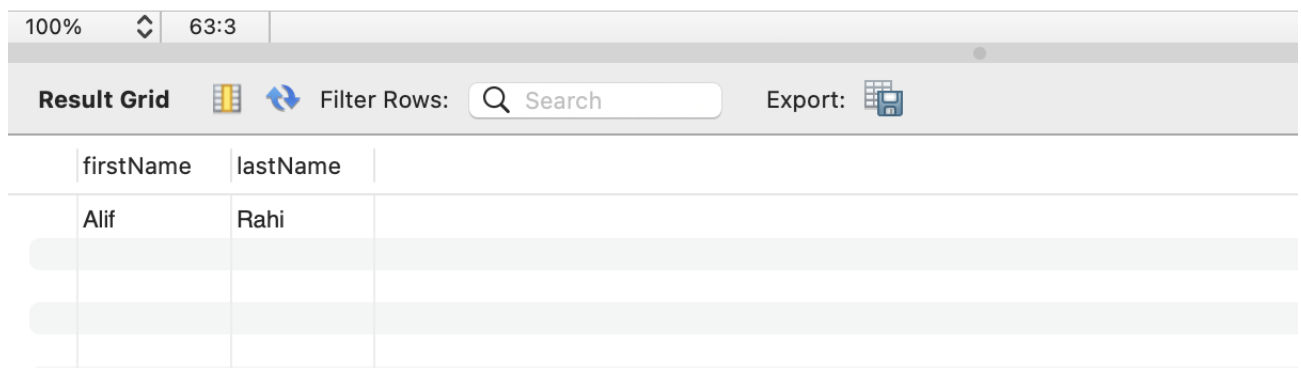
```
1 • UPDATE mission SET
2   status = 'Canceled' WHERE
3   missionName = 'STS-7';
```

8. Identify astronauts without missions. Display the astronaut name. Use a nested select to answer this question.



The screenshot shows a SQL editor with a toolbar at the top. A search bar on the right indicates 'Limit to 50000 rows'. Below the toolbar is a 'Find' and 'Replace' section. The main area contains the following SQL query:

```
1 • SELECT DISTINCT firstName, lastName
2   FROM astronauts, missions_astro WHERE astronauts.Id
3   NOT IN (SELECT missions_astro.astronautId FROM missions_astro)
```

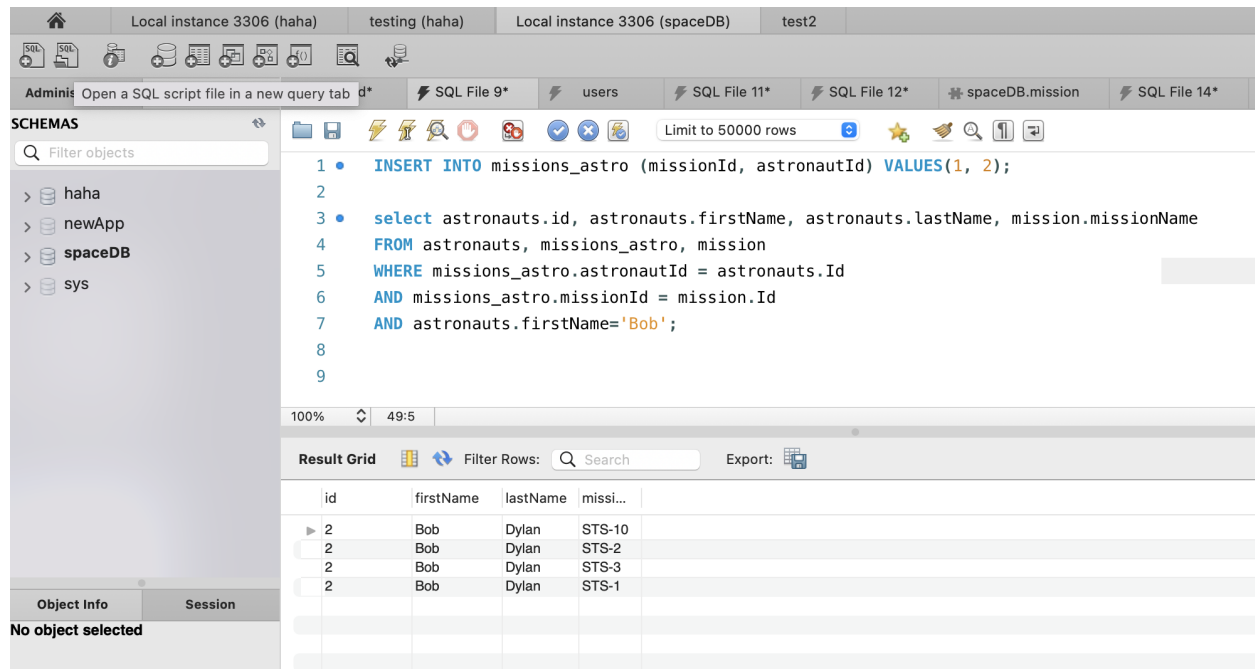


The screenshot shows a 'Result Grid' with a toolbar at the top. The toolbar includes a 'Filter Rows' section with a search bar and an 'Export' button. The grid displays the results of the SQL query, showing the first and last names of astronauts without missions.

firstName	lastName
Alif	Rahi

9. In one SQL window, reassign Astronaut A to mission 1. Don't commit. In another SQL window, delete mission 1. Don't commit. Explain your results. Resolve the problem.

(Table before **committing** in root users view)



The screenshot shows a SQL IDE interface. The top bar indicates the current instance is 'Local instance 3306 (haha)'. The left sidebar shows the 'SCHEMAS' panel with a tree view containing 'haha', 'newApp', 'spaceDB', and 'sys'. The main query window displays the following SQL code:

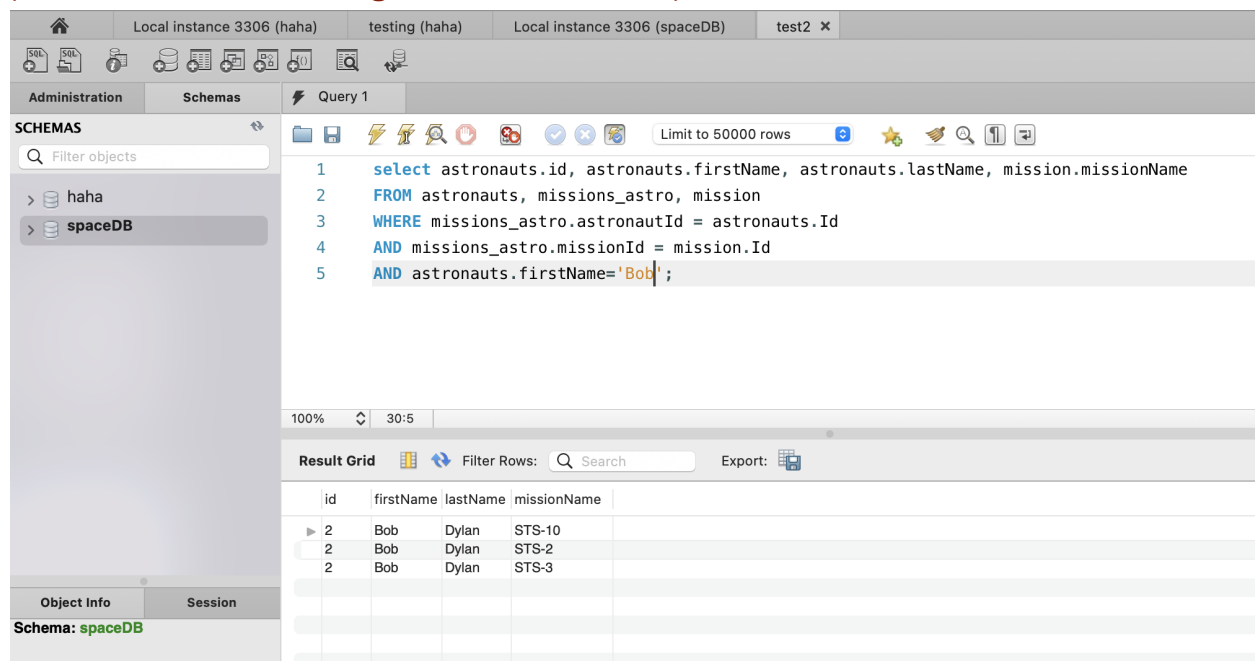
```
1 • INSERT INTO missions_astro (missionId, astronautId) VALUES(1, 2);
2
3 • select astronauts.id, astronauts.firstName, astronauts.lastName, mission.missionName
4 FROM astronauts, missions_astro, mission
5 WHERE missions_astro.astronautId = astronauts.Id
6 AND missions_astro.missionId = mission.Id
7 AND astronauts.firstName='Bob';
8
9
```

Below the query window, the 'Result Grid' is visible, showing 4 rows of data:

id	firstName	lastName	missionName
2	Bob	Dylan	STS-10
2	Bob	Dylan	STS-2
2	Bob	Dylan	STS-3
2	Bob	Dylan	STS-1

The status bar at the bottom indicates 'No object selected'.

(Table before **committing** in test2 users view)



The screenshot shows the same SQL IDE interface, but the current instance is 'test2'. The left sidebar shows the 'SCHEMAS' panel with a tree view containing 'haha' and 'spaceDB'. The main query window displays the following SQL code:

```
1 select astronauts.id, astronauts.firstName, astronauts.lastName, mission.missionName
2 FROM astronauts, missions_astro, mission
3 WHERE missions_astro.astronautId = astronauts.Id
4 AND missions_astro.missionId = mission.Id
5 AND astronauts.firstName='Bob';
```

Below the query window, the 'Result Grid' is visible, showing 3 rows of data:

id	firstName	lastName	missionName
2	Bob	Dylan	STS-10
2	Bob	Dylan	STS-2
2	Bob	Dylan	STS-3

The status bar at the bottom indicates 'Schema: spaceDB'.

(Table after **committing** in root users view)

SQL Developer interface showing a query execution in the root user view. The query is:

```

1 • select astronauts.id, astronauts.firstName, astronauts.lastName, mission.missionName
2 FROM astronauts, missions_astro, mission
3 WHERE missions_astro.astronautId = astronauts.Id
4 AND missions_astro.missionId = mission.Id
5 AND astronauts.firstName='Bob';
6
7 • commit;
8

```

The result grid shows the following data:

id	firstName	lastName	missionName
2	Bob	Dylan	STS-10
2	Bob	Dylan	STS-2
2	Bob	Dylan	STS-3
2	Bob	Dylan	STS-1

(Table after **committing** in test2 users view)

SQL Developer interface showing a query execution in the test2 user view. The query is:

```

1 select astronauts.id, astronauts.firstName, astronauts.lastName, mission.missionName
2 FROM astronauts, missions_astro, mission
3 WHERE missions_astro.astronautId = astronauts.Id
4 AND missions_astro.missionId = mission.Id
5 AND astronauts.firstName='Bob';

```

The result grid shows the following data:

id	firstName	lastName	missionName
2	Bob	Dylan	STS-10
2	Bob	Dylan	STS-2
2	Bob	Dylan	STS-3
2	Bob	Dylan	STS-1

- When I assigned mission 1 to Astronaut 'Bob', other people couldn't see it and only saw the **committed** changes. In this case, even if i **deleted** or added new entries into my table it won't be shown in the 'test2' users account because I didn't **commit** it from the root user yet. After I commit, you can see the updated table with **mission 1** added.

10. In one SQL window, delete the astronaut Sally Ride. Don't commit. In another SQL window, reassign Sally Ride to a new mission. Don't commit. Explain your results. Resolve the problem.

(Here we **delete** Sally from our DB but we don't commit yet)

The screenshot shows the SQL Developer interface. The 'Schemas' pane on the left shows the 'spaceDB' schema with tables 'astronauts', 'mission', 'missions\_astro', 'objectives', and 'spaceship'. The main query window shows the following SQL code:

```
1 • DELETE from astronauts WHERE firstName= 'Sally'
2   AND lastName = 'Ride';
3 • SELECT * FROM astronauts;
```

The 'Result Grid' shows the data from the 'astronauts' table. The table has columns: id, firstName, lastName, country, gender, DOB, and degree. The data is as follows:

id	firstName	lastName	country	gender	DOB	degree
1	Zack	Hermin	USA	M	1966-03-11	Computer Science
2	Bob	Dylan	USA	M	1964-05-01	Music
3	Mark	Beltron	Algeria	M	1956-03-12	Economics
4	Koby	White	Russia	M	1976-01-17	Economics
5	Malissa	Shiffman	USA	F	1977-01-14	Computer Science
6	Courtney	Dill	USA	F	1959-02-13	Physics
8	Presh	Takur	India	M	1956-06-12	Computer Science
9	John	Smith	USA	M	1969-09-07	Music
10	William	lin	USA	M	1976-01-11	Physics
11	Alif	Rahi	USA	M	2000-06-29	Computer Science
12	NULL	NULL	NULL	NULL	NULL	NULL

(We just deleted 'Sally' in the root user account, so why do we still see it on id=12?  
..Because the transaction was not committed)

The screenshot shows the SQL Developer interface. The 'Schemas' pane on the left shows the 'spaceDB' schema with tables 'astronauts', 'mission', 'missions\_astro', 'objectives', and 'spaceship'. The main query window shows the following SQL code:

```
1 • SELECT * from astronauts;
```

The 'Result Grid' shows the data from the 'astronauts' table. The table has columns: id, firstName, lastName, country, gender, DOB, and degree. The data is as follows:

id	firstName	lastName	country	gender	DOB	degree
1	Zack	Hermin	USA	M	1966-03-11	Computer Science
2	Bob	Dylan	USA	M	1964-05-01	Music
3	Mark	Beltron	Algeria	M	1956-03-12	Economics
4	Koby	White	Russia	M	1976-01-17	Economics
5	Malissa	Shiffman	USA	F	1977-01-14	Computer Science
6	Courtney	Dill	USA	F	1959-02-13	Physics
8	Presh	Takur	India	M	1956-06-12	Computer Science
9	John	Smith	USA	M	1969-09-07	Music
10	William	lin	USA	M	1976-01-11	Physics
11	Alif	Rahi	USA	M	2000-06-29	Computer Science
12	Sally	Ride	USA	F	1982-11-29	Math
13	NULL	NULL	NULL	NULL	NULL	NULL

(The 'test2' user doesn't know about Sally getting **deleted** so it is able to **assign** new mission Entries in the mission\_astro table for Sally)

The screenshot shows a database management interface with a top toolbar, a left sidebar for 'SCHEMAS', a central query editor, and a bottom 'Result Grid'.

**SCHEMAS Sidebar:**

- haha
- spaceDB
  - Tables
    - astronauts
    - mission
    - missions\_astro
    - objectives
    - spaceship
  - Views
  - Stored Procedures
  - Functions

**Query Editor:**

```
1 • INSERT INTO missions_astro (missionId, astronautId) VALUES
2   (1,12),
3   (5,12),
4   (2,12),
5   (9,12);
6 • SELECT astronauts.id, astronauts.firstName, astronauts.lastName, mission.missionName
7   FROM astronauts, missions_astro, mission
8   WHERE missions_astro.astronautId = astronauts.Id
9   AND missions_astro.missionId = mission.Id
10  AND astronauts.firstName='Sally';
11
```

**Result Grid:**

	id	firstName	lastName	missionName
▶	12	Sally	Ride	STS-1
	12	Sally	Ride	STS-5
	12	Sally	Ride	STS-2
	12	Sally	Ride	STS-9

- We can easily resolve this in the root account by doing one of the 2 things:
  - a **Rollback** (If we wan't Sally back).
  - A **Commit** (If we want to keep Sally deleted).



11. In one SQL window, rename the Space Shuttle Enterprise to Voyager. Don't commit. In another SQL window, change the Space Shuttle Enterprise to Lexington. Don't commit. Quit both Oracle sessions. Login to Oracle and display all information for the Space Shuttle Enterprise . Explain your results.

- The Space Shuttle Enterprise stayed as it's pre-committed value and did not change to **Voyager**. This is due to the fact that again, we did not commit and decided to log out. This will automatically **Rollback** to the most recently committed Data.

12)

**Table: Astronauts**



1	DESC astronauts
---	-----------------

100%

16:1

Result Grid

Filter Rows:

Search

Export:

	Field	Type	Null	Key	Default	Extra
	id	int	NO	PRI	NULL	auto_increment
	firstName	varchar(50)	YES		NULL	
	lastName	varchar(50)	YES		NULL	
	country	varchar(50)	YES		NULL	
	gender	char(1)	YES		NULL	
	DOB	date	YES		NULL	
	degree	varchar(50)	YES		NULL	

**Table: Mission**

Limit to 50000 rows

1DESC mission

100%

13:1

Result Grid

Filter Rows:

Export:

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	missionName	varchar(30)	YES		NULL	
	launchSight	varchar(50)	YES		NULL	
	landingSight	varchar(50)	YES		NULL	
	duration	bigint	YES		NULL	
	spaceshipId	int	YES	MUL	NULL	
	sightCountry	varchar(20)	YES		NULL	
	sightState	varchar(20)	YES		NULL	
	startDate	date	YES		NULL	
	endDate	date	YES		NULL	
	mileage	int	YES		NULL	
	status	varchar(20)	YES		NULL	

**Table: mission\_astro**

Limit to 50

1DESCmissions\_astro

100%

20:1











Result Grid

Filter Rows:

Export

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	missionId	int	YES	MUL	NULL	
	astronautId	int	YES	MUL	NULL	

### Table: Objectives

										Limit to 5000
1	DESC objectives									

100%

16:1

Result Grid

Filter Rows:

Search

Export:

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
◀	missionId	int	YES	MUL	NULL	
	objective	varchar(100)	YES		NULL	

### Table: Spaceships

Limit to 50000 rows

1DESC spaceship

100%

15:1

Result Grid

Filter Rows:

Search

Export:

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	<div>NULL</div>	auto_increment
◀	name	varchar(50)	YES		<div>NULL</div>	
	Fate	varchar(20)	YES		<div>NULL</div>	
◀	Manufacturer	varchar(50)	YES		<div>NULL</div>	