SmartForm Database User Manual

Contents

1: Introduction

SmartForm Database Overview

The SmartForm database is a comprehensive source of horseracing form for races run in the UK and Ireland consisting of two main components:

1. **The SmartForm Historic database**, which contains over 5 years' data for horseracing results in the UK and Ireland, detailing race and runner attributes, from January 1st 2003 to the date of purchase, from October 2008 onwards.

2. **The SmartForm Daily service**, which brings the historic database up to date beyond the date of purchase, up to and including the current date, for current subscribers, and supplies results and racecards on a daily basis, specifically:

    a. Automatically updating the historic database with the previous day's results
    b. Automatically updating the database with daily racecards prior to racing, consisting of race and runner data for all UK and Irish meetings, accessible programmatically.

The historic data can be used to perform research and analysis, to test out potential betting systems and strategies, and to build predictive models or ratings to assess comparative chances within a race.

The daily data enables the user to apply systems and predictive models to each race for the day ahead. The data from the daily racecards contains runner and past performance information that can also be used to apply existing betting systems, often without daily reference to the historic database for previous form.

Used together, both components enable bettors to analyse and produce models for UK and Irish horseracing that are tested on up to the minute data and then to apply them to daily races. Since the database is configured for use with MySQL, such a process can be automated and integrated within an automatic betting strategy, such as outlined in the book *Automatic Exchange Betting*.

Note that the SmartForm data, database formats, and documentation are licenced for personal use only. The data and database documentation is not for redistribution, publication or resale.

Database structure

The structure of the database is simple, with 4 tables in total.

For the **historic database**, there are 2 tables:

historic_runners
historic_races

All runners are linked to races by the **race_id** field which is common to both tables. All the fields available within each table are described in Section 2. Examples of using the historic database begin in Section 3.

The **SmartForm Daily** service updates the historic tables automatically with all results in the UK and Ireland, maintaining them up to the current date.

For the daily racecards provided by the **SmartForm Daily** service, there are two more tables:

daily_runners
daily_races

Daily runners are linked to daily races by the **race_id** field. All fields available within each table are detailed in Section 2. Examples of using the database tables for daily racecards begin in Section 4.

Database Management System - MySQL

The SmartForm database download and the Smartform Daily service are designed for use with the MySQL database management system (DBMS). MySQL is one of the most robust and popular databases in the world, that runs on Windows, Mac and Linux machines, is available for free download at www.mysql.com.

Installing MySQL is a prerequisite for using the SmartForm database, although data can subsequently be exported to other formats if required.

This user guide provides examples showing how to use the SmartForm database in the MySQL database management system, but it is no substitute for understanding how MySQL works – the reference materials referred to in Appendix 1 are recommended for this purpose.

2: Installation of the SmartForm Database

This section covers installation of the database, assuming that the database has been successfully downloaded from www.betwise.co.uk

Notes on installing MySQL are also provided when the database is downloaded. If you have followed the installation process from the website, you can ignore this section.

Email support on database installation is also available at www.betwise.co.uk/contact.

This section therefore provides a summary of the installation process.

Installing on Windows

1. Install mysql at www.mysql.com/downloads Choose Community Edition (the free download) and the correct version for your operating system.

2. Create a Smartform database, for example:

a) Type the following to log into the mysql monitor at the command prompt (> below):

```
>mysql –u root –p
```

b) You should now enter your password, provided that one was set up during the installation of MySQL

```
        >Enter password:
```

Whereupon you will successfully be logged in

```
        Welcome to the MySQL monitor.  Commands end with ; or \g.
```

c) Next, set up the database access as follows:

```
>CREATE DATABASE smartform;
```

Then, grant access to your user name for the Smartform database.

```
>GRANT ALL ON smartform.* TO ‘your_account_name’@’localhost’;
```

Log off from the MySQL monitor to return to the shell prompt:

```
>exit;
```

3. Download the smartform data, extracted to the C:\Smartform directory and load it into the database. Step 4 explains how.

4. Provided that the database download file has been unzipped to smartform.ql, building the database can be done from the command line as follows:

```
>mysql –u smartform – p [insert password  if you set one up] < smartform.sql
```

From now on you can log yourself in by the specified username to interact with the database, or specify the same user credentials within the context of a program to access the database.

Installing Daily Automated Updates Scripts on Windows

Nb. Steps 5 onwards only apply if you are a subscriber to the Daily Updates Service

5. Download the scripts.zip file and unpack it

6. The updates script is called fetch_updates.bat in the windows/ directory within the scripts package.

7. Create a directory, say C:\smartform

8. Store the batch file in this directory

9. Download curl (a command-line internet file fetching utility) from:

   http://curl.haxx.se/download.html

   (pick a Win32 version appropriate to your system)

10. Unpack the curl archive, and save curl.exe somewhere, perhaps in C:\smartform

11. Edit the batch file to specify the username, password, database name and optionally database hostname of the DB where you are storing the smartform data

12. Edit the script to specify your betwise user ID and download key

13. Check that the paths to mysql and curl specified in the batch file are correct, or edit them if not

14. To update the daily racecards, run fetch_updates.bat with the parameter: daily

   e.g. C:\smartform\fetch_updates.bat daily

- similarly, to update the historic race data, run fetch_updates.sh with the parameter: historic

   e.g. C:\smartform\fetch_updates.bat historic

15. To automate these tasks, you can set them up as windows Scheduled Tasks:

- at the windows prompt, enter:

```
 at 00:30 /every:monday,tuesday,wednesday,thursday,friday,saturday,sunday
c:\smartform\fetch_updates.bat daily
```

```
 at 05:30 /every:monday,tuesday,wednesday,thursday,friday,saturday,sunday
c:\smartform\fetch_updates.bat historic
```

- you can opt not to run the scheduler every day.

- if you wish to view these jobs, you can do so from the Scheduled Tasks tool:

  go to Start > All Programs > Accessories > System Tools > Scheduled Tasks

- If you have not used the 'at' utility before, these jobs will be called At1 and At2. You may wish to rename them.

- If running the scripts would require you to be logged on (for example to make use of a shared drive), you may need to manually configure the tasks using the Scheduled Tasks utility.

Installing on Linux and Mac OS X Operating Systems

1. Install mysql at www.mysql.com/downloads Choose Community Edition (the free download) and the correct version for your operating system.

2. Create a Smartform database, for example:

a) Type the following to log into the mysql monitor at the command prompt (> below):

```
>mysql –u root –p
```

b) You should now enter your password, provided that one was set up during the installation of MySQL

```
        >Enter password:
```

Whereupon you will successfully be logged in

```
        Welcome to the MySQL monitor.  Commands end with ; or \g.
```

c) Next, set up the database access as follows:

```
>CREATE DATABASE smartform;
```

Then, grant access to your user name for the Smartform database.

```
>GRANT ALL ON smartform.* TO ‘your_account_name’@’localhost’;
```

Log off from the MySQL monitor to return to the shell prompt:

```
>exit;
```

4. Download the smartform data, extracted to the C:\Smartform directory and load it into the database. Step 4 explains how.

4. Provided that the database download file has been unzipped to smartform.ql, building the database can be done from the command line as follows:

```
>mysql –u smartform – p [insert password  if you set one up] < smartform.sql
```

From now on you can log yourself in by the specified username to interact with the database, or specify the same user credentials within the context of a program to access the database.

Installing Daily Automated Updates Scripts on Linux and Mac OS X Operating Systems

Nb. Steps 5 onwards only apply if you are a subscriber to the Daily Updates Service

5. Download the scripts.zip file and unpack it

6. The updates script is called fetch_updates.sh in the unix/ directory within the scripts package.

7. Install this script file somewhere (perhaps in a bin/ directory in your home directory)

8. Edit the script file to specify the username, password, database name and optionally database hostname of the DB where you are storing the smartform data

9. Edit the script to specify your betwise user ID and download key

10. To update the daily racecards, run fetch_updates.sh with the parameter: daily

```
e.g. ~/bin/fetch_updates.sh daily
```

- similarly, to update the historic race data, run fetch_updates.sh with the parameter: historic

```
e.g. ~/bin/fetch_updates.sh historic
```

11. To automate these updates, create a crontab entry for each. Edit your crontab with the command:

```
crontab -e
```

and add these lines:

```
# fetch the racecards at half past midnight
30 00 * * * /home/smartform/bin/fetch_updates.sh daily

# fetch the result files at 5:30am
30 05 * * * /home/smartform/bin/fetch_updates.sh historic
```

3: SmartForm Database Tables and values

There are 4 basic tables that come as standard in the database, each of which is automatically updated by the Daily Updates service:

- **historic_races**
- **historic_runners**
- **daily_races**
- **daily_runners**

Each table is listed with each field name that is found within the table in the subsequent sections of this Section. For each field, there is a description of what is represented, the MySQL data type of the field, and whether or not the field can hold a null value.

Historic Races Table: historic_races

Field	Description	Data Type	Null
race_id	internal field, reference for race_id field, corresponds to historic_runners table, PRIMARY KEY	int(11)	NO
meeting_id	internal database field	int(11)	NO
meeting_date	date of the meeting and therefore the race in question	date	NO
course	name of course	varchar(255)	NO
conditions	conditions of race, eg. Handicap Chase 0-125	varchar(255)	NO
race_name	Name of the race	varchar(255)	NO
race_abbrev_name	same data as conditions field, slightly different format (ages in brackets)	varchar(80)	NO
race_type_id	internal field referencing race_type by a number, useful for queries on certain types	int(11)	NO
race_type	type of race, values being Flat, Hurdle, Chase, NH Flat, AW Flat, Point to Point	varchar(80)	NO
race_num	number of race on racecard in terms of running order (eg. first race = 1)	tinyint(2)	NO
going	going, eg. Hard, Firm, Good to Firm, Good, Good to Soft, Standard etc.	varchar(80)	YES
direction	Direction of track, eg. Left Handed, Right Handed, Straight	varchar(80)	YES
class	Class of race on flat, by number, entries prior to change in system in 2005 are irrelevant.	tinyint(2)	YES
draw_advantage	comment on draw advantage for meeting, eg "Low best in races up to a mile"	varchar(80)	YES

historic_races table continued…

Field	Description	Data Type	Null
num_fences	number of fences in race if applicable race type, otherwise zero or NULL	tinyint(2)	YES
handicap	if handicap race, 0 or 1 (1 if true)	tinyint(1)	YES
all_weather	if all weather race, 0 or 1 (1 if true)	tinyint(1)	YES
seller	if selling race, 0 or 1 (1 if true)	tinyint(1)	YES
claimer	if claimer race, 0 or 1 (1 if true)	tinyint(1)	YES
apprentice	if apprentice race, 0 or 1 (1 if true)	tinyint(1)	YES
maiden	if maiden race, 0 or 1 (1 if true)	tinyint(1)	YES
amateur	if amateur race, 0 or 1 (1 if true)	tinyint(1)	YES
num_runners	number of runners declared on day of race, can be unreliable due to earlier declarations counted	tinyint(2)	YES
num_finishers	number of runners who completed race	tinyint(2)	YES
rating	if a handicap race, the upper rating bound, eg. 0-125 handicap returns 125 here	int(11)	YES
group_race	if a group race (or graded race in National Hunt), the rank of the race, ie. 1, 2 or 3	int(11)	YES
min_age	the minimum age eligible to compete in race if captured (see also conditions)	tinyint(2)	YES
max_age	the maximum age eligible to compete in race if captured (see also conditions)	tinyint(2)	YES
distance_yards	the race distance in yards	int(11)	YES
added_money	total prizemoney for the race	float(8,2)	YES
official_rating	official rating, placeholder field	int(11)	YES
speed_rating	speed_rating for race, placeholder field	int(11)	YES
private_handicap	private handicap for race, placeholder field	int(11)	YES
scheduled_time	displays the schedule date and time for the race, in yyyy/mm/dd hh:mm:ss format	datetime	NO
off_time	displays the actual time that the race started, using date field and 12 hour clock, needs transforming	datetime	NO
winning_time_disp	The winning time in minutes:seconds:milliseconds format	varchar(20)	YES
winning_time_secs	The winning time in seconds decimal format only	float(10,2)	YES

historic_races table continued…

Field	Description	Data Type	Null
standard_time_disp	the standard time for this race and distance in minutes:seconds:milliseconds format	varchar(20)	YES
standard_time_secs	the standard time for this race in seconds decimal format	float(10,2)	YES
loaded_at	internal smartform field	timestamp	NO

Historic Runners Table: historic_runners

Field	Description	Data Type	Null
runner_id	internal field, reference for each unique runner, PRIMARY KEY	int(11)	NO
race_id	internal field, reference for race_id field, corresponds to historic_races table, PRIMARY KEY	int(11)	NO
name	name of horse	varchar(255)	NO
foaling_date	horse date born	date	NO
colour	colour eg. Bay, chestnut, etc	varchar(20)	NO
distance_travelled	distance travelled from stable to racecourse	int(11)	YES
form_figures	string of positions in previous races	varchar(80)	YES
gender	Letter representing sex of horse (G)elding, (F)illie, (M)are, (C)olt, (H)orse	char(1)	YES
age	age of horse in years	tinyint(3)	YES
bred	country of breeding represented by 2-3 letter code in capitals, eg. IRE, FR, UK	char(3)	YES
cloth_number	saddlecloth number for race	tinyint(2)	YES
stall_number	number drawn in stalls	tinyint(2)	YES
num_fences_jumped	number of fences jumped if relevant	tinyint(2)	YES
long_handicap	number of pounds carried over official handicap mark (ie. number of pounds out of handicap)	int(11)	YES
how_easy_won	placeholder field	int(11)	YES
in_race_comment	in running comment for the horse in the race	text	YES
official_rating	official rating for horse performance in race, shown where available	int(11)	YES

historic_runners table continued…

Field	Description	Data Type	Null
official_rating_type	official rating for horse performance in race, shown where available	int(11)	YES
speed_rating	private speed rating for information purposes, shown where available	int(11)	YES
speed_rating_type	private speed rating for information purposes, shown where available	int(11)	YES
private_handicap	private handicap rating for information purposes, shown where available	int(11)	YES
private_handicap_type	private handicap rating for information purposes, shown where available	int(11)	YES
trainer_name	name of trainer	varchar(80)	YES
trainer_id	internal field for trainer_id reference	int(11)	YES
owner_name	name of owner	varchar(80)	YES
owner_id	internal field for owner_id reference	int(11)	YES
jockey_name	name of jockey	varchar(80)	YES
jockey_id	internal field for jockey_id reference	int(11)	YES
jockey_claim	jockey claim in pounds if applicable	int(3)	YES
dam_name	name of horse's dam	varchar(80)	YES
dam_id	internal field for dam_id reference	int(11)	YES
sire_name	name of horse's sire	varchar(80)	YES
sire_id	internal field for sire_id reference	int(11)	YES
dam_sire_name	name of dam's sire	varchar(80)	YES
dam_sire_id	internal field for dam_sire_id reference	int(11)	YES
forecast_price	the forecast price from racecard as odds to one in character format	char(10)	YES
forecast_price_decimal	the forecast price from racecard in decimal format	float(8,2)	YES
starting_price	the starting price as odds to one in character format	char(10)	YES
starting_price_decimal	the starting price in decimal format	float(8,2)	YES
betting_text	any comments about pre-race price movements, eg. "op 7/2 tchd 10/3 and 13/2"	text	YES
position_in_betting	rank of runner in starting prices returned from the race, where 1 = favourite	tinyint(3)	YES

historic_runners table continued…

finish_position	the finishing position of the horse in the race	tinyint(3)	YES
amended_position	the amended position of the horse in the race, if relevant, either zero or amended position	tinyint(3)	YES
unfinished	if the horse did not finish the race, the reason for that, including "Nonrunner".	varchar(30)	YES
distance_beaten	distance the horse was beaten by the horse immediately in front of it, in lengths	float(8,2)	YES
distance_won	if the horse won, how far the horse won by, in lengths	float(8,2)	YES
distance_behind_winner	distance the horse was beaten by the winner of the race, in lengths	float(8,2)	YES
prize_money	prize money received from race if applicable in GBP	float(8,2)	YES
tote_win	price paid for the tote win for this horse, if applicable	float(8,2)	YES
tote_place	price paid for the tote place for this horse, if applicable	float(8,2)	YES
days_since_ran	number of days since the horse last ran	int(4)	YES
last_race_type	type of race that the horse last competed in	varchar(80)	YES
last_race_type_id	type_id of the race that the horse last competed in	int(11)	YES
last_race_beaten_fav	whether or not a beaten favourite in the last race, 1 if a beaten favourite, 0 if not, NULL if N/A	int(1)	YES
weight_pounds	weight carried in pounds	int(11)	YES
penalty_weight	penalty weight carried in pounds if applicable	int(11)	YES
over_weight	any over weight in pounds, above the official weight that the horse is allocated	int(11)	YES
tack_hood	whether or not a hood was worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
tack_visor	whether or not a visor was worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
tack_blinkers	whether or not blinkers were worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
tack_eye_shield	whether or not eye shields were worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
tack_eye_cover	whether or not an eye cover was worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
tack_cheek_piece	whether or not cheek pieces were worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
tack_pacifiers	whether or not pacifiers were worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
tack_tongue_strap	whether or not a tongue strap was worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
loaded_at	internal smartform field	timestamp	NO

Daily Races Table: daily_races

Field	Description	Data Type	Null
race_id	internal field, reference for race_id field, corresponds to historic_runners table, PRIMARY KEY	int(11)	NO
meeting_id	internal database field	int(11)	NO
meeting_date	date of the meeting and therefore the race in question	date	NO
weather	text string for weather forecast	text	YES
meeting_status	smartform internal field	varchar(80)	YES
meeting_abandoned_reason	smartform internal field	varchar(80)	YES
draw_advantage	comment on draw advantage for meeting, eg "Low best in races up to a mile"	text	YES
course	name of course	varchar(255)	NO
country	country of race meeting	varchar(80)	YES
race_title	name of the race	varchar(255)	NO
race_type	type of race, values being Flat, Hurdle, Chase, NH Flat, AW Flat, Point to Point	varchar(80)	YES
track_type	Direction of track, eg. Left Handed, Right Handed, Straight	varchar(80)	YES
advanced_going	forecast going, eg. Hard, Firm, Good to Firm, Good, Good to Soft, Standard etc.	varchar(80)	YES
class	Class of race on flat, by number, entries prior to change in system in 2005 are irrelevant.	varchar(80)	YES
handicap	if handicap race, 0 or 1 (1 if true)	tinyint(1)	YES
trifecta	whether or not race is a trifecta race, 0 or 1 (1 if true)	tinyint(1)	YES
showcase	whether or not race is a showcase race, 0 or 1 (1 if true)	tinyint(1)	YES
age_range	description of range of eligible ages for race, eg. "4YO to 6YO" or "2YO only". All start "#YO"	varchar(30)	YES
distance_yards	the race distance in yards	int(11)	YES
added_money	total prizemoney for the race	float(8,2)	YES
penalty_value	the prize money that the winner will be deemed to have won	float(8,2)	YES
scheduled_time	time that race is due off, including the date in datetime format, so yyyy-mm-dd hh:mm:ss	datetime	NO

daily_races table continued…

Field	Description	Data Type	Null
prize_pos_1	prize money for first place	float(8,2)	YES
prize_pos_2	prize money for second place, where available	float(8,2)	YES
prize_pos_3	prize money for third place, where available	float(8,2)	YES
prize_pos_4	prize money for fourth place, where available	float(8,2)	YES
prize_pos_5	prize money for fifth place, where available	float(8,2)	YES
prize_pos_6	prize money for sixth place, where available	float(8,2)	YES
prize_pos_7	prize money for seventh place, where available	float(8,2)	YES
prize_pos_8	prize money for eighth place, where available	float(8,2)	YES
prize_pos_9	prize money for ninth place, where available	float(8,2)	YES
last_winner_no_race	if a corresponding race to this happened last year, this field is NULL, otherwise reason is shown	varchar(255)	YES
last_winner_year	the year referred to by "last_winner_*" fields	int(4)	YES
last_winner_runners	the number of runners in the corresponding race last year	int(3)	YES
last_winner_runner_id	the runner_id for winner of the corresponding race last year	int(11)	YES
last_winner_name	the name of the winner in the corresponding race last year	varchar(255)	YES
last_winner_age	the age of the winner in the corresponding race last year	tinyint(2)	YES
last_winner_bred	the country of breeding of the winner in the corresponding race last year	char(3)	YES
last_winner_weight	the weight of the winner in the corresponding race last year	int(11)	YES
last_winner_trainer	the winning trainer from the corresponding race last year	varchar(80)	YES
last_winner_trainer_id	the trainer_id of the winning trainer in the corresponding race last year	int(11)	YES
last_winner_jockey	the winning jockey from the corresponding race last year	varchar(80)	YES
last_winner_jockey_id	the trainer_id of the winning jockey in the corresponding race last year	int(11)	YES
last_winner_sp	the starting price of the winner from the corresponding race last year	char(10)	YES
last_winner_sp_decimal	the SP in decimal odds of the winner in the corresponding race last year	float(8,2)	YES

daily_races table continued…

Field	Description	Data Type	Null
last_winner_betting_ranking	the ranking in the betting (eg. 1 = FAV) of the winner in the corresponding race last year	int(3)	YES
last_winner_course_winner	whether the winner of the corresponding race last year was a previous course winner	tinyint(2)	YES
last_winner_distance_winner	whether the winner of the corresponding race last year was a previous distance winner	tinyint(2)	YES
last_winner_candd_winner	whether the winner of the corresponding race last year was a previous course and distance winner	tinyint(2)	YES
last_winner_beaten_favourite	whether the winner of the corresponding race last year was a previous beaten favourite	tinyint(2)	YES
loaded_at	smartform internal field	timestamp	NO

Daily Runners Table: daily_runners

Field	Description	Data Type	Null
runner_id	internal field, reference for each unique runner, PRIMARY KEY	int(11)	NO
race_id	internal field, reference for race_id field, corresponds to daily_races table, PRIMARY KEY	int(11)	NO
name	name of horse	varchar(255)	NO
foaling_date	horse date born	date	YES
age	age of horse in years	tinyint(2)	YES
colour	colour eg. Bay, chestnut, etc	varchar(20)	YES
form_figures	string of positions in previous races	varchar(80)	YES
form_type	type of race for last run races	varchar(30)	YES
gender	Letter representing sex of horse (G)elding, (F)illie, (M)are, (C)olt, (H)orse	char(1)	YES
bred	country of breeding represented by 2-3 letter code in capitals, eg. IRE, FR, UK	char(3)	YES
cloth_number	saddlecloth number for race	tinyint(2)	YES
stall_number	number drawn in stalls	tinyint(2)	YES

daily_runners table continued…

Field	Description	Data Type	Null
long_handicap	number of pounds carried over official handicap mark (ie. number of pounds out of handicap)	int(11)	YES
official_rating	the official rating of the horse for today's contest	int(11)	YES
adjusted_rating	the adjusted rating of the horse for today's contest	int(11)	YES
trainer_name	name of trainer	varchar(80)	YES
trainer_id	internal field for trainer_id reference	int(11)	YES
owner_name	name of owner	varchar(80)	YES
jockey_name	internal field for owner_id reference	varchar(80)	YES
jockey_id	name of jockey	int(11)	YES
jockey_claim	claim of jockey if any	int(3)	YES
jockey_colours	jockey colours	varchar(255)	YES
dam_name	name of horse's dam	varchar(80)	YES
dam_year_born	year dam was born	smallint(4)	YES
sire_name	name of horse's sire	varchar(80)	YES
sire_year_born	year sire was born	smallint(4)	YES
dam_sire_name	name of dam's sire	varchar(80)	YES
dam_sire_year_born	year dam's sire was born	smallint(4)	YES
forecast_price	the forecast price from racecard as odds to one in character format	char(10)	YES
forecast_price_decimal	the forecast price from racecard in decimal odds format	float(8,2)	YES
days_since_ran	number of days since horse last ran	int(4)	YES
days_since_ran_type	the type of race the horse last ran in	varchar(30)	YES
weight_pounds	weight carried in pounds	int(11)	YES
tack_hood	whether or not a hood was worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
tack_visor	whether or not a visor was worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
tack_blinkers	whether or not blinkers were worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES

daily_runners table continued…

Field	Description	Data Type	Null
tack_eye_shield	whether or not eye shields were worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
tack_eye_cover	whether or not an eye cover was worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
tack_cheek_piece	whether or not cheek pieces were worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
tack_pacifiers	whether or not pacifiers were worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
tack_tongue_strap	whether or not a tongue strap was worn, NULL or 1 (to indicate this tack present)	tinyint(1)	YES
course_winner	how many times course winner, ie. Number of times or NULL	tinyint(2)	YES
distance_winner	how many times distance winner ie. Number of times or NULL	tinyint(2)	YES
candd_winner	how many times course and distance winner ie. Number of times or NULL	tinyint(2)	YES
beaten_favourite	if beaten favourite last time out, how many times ever beaten favourite, ie. Number of times or NULL	tinyint(2)	YES
loaded_at	smartform internal field	timestamp	NO

4: Using The SmartForm Database

Usage Overview

The SmartForm database provides UK and Irish racing data within the context of a powerful programming environment.

Since the database is designed for use with MySQL, all the facilities within MySQL, (including the power of Structured Query Language (SQL) for interactive queries, and all the programmatic interfaces to MySQL, such as Perl DBI) are available to users for manipulating and querying the SmartForm database. Users who are skilled in SQL can “ask questions” and get back answers from the data.

As such, there is no **prescriptive** way to use the database, there are, for all practical purposes, infinite ways to use and program the SmartForm database. The ways in which the database is used will depend on what the user wants to do, their understanding of the fundamental variables used, and their knowledge of the database management system. This manual provides a start for new users by discussing some typical uses, describing the fundamental variables, and showing examples via the MySQL command line client that give an indication of how MySQL can be used to explore the database.

For any SmartForm users who are unfamiliar with MySQL or SQL, the best place to start in order to get the most out of the SmartForm database is with a guide to MySQL or SQL itself (and using the query language in particular, since the database is already there). Some of the best references for getting started with MySQL are listed in Appendix 1. Knowledge of SQL should be combined with an understanding of the different variables within the database, as listed in Section 2.

Therefore, this Section provides some specific examples to show how the database can typically be used (which can easily be copied and extended for more general uses), but is no substitute for references to MySQL and structured query language (SQL) in general.

Querying Historic Tables

This section runs through some usage examples from the historic database tables in order to illustrate common database operations.

All the database operations are shown from the perspective of using the interactive MySQL command line client that comes as standard with the MySQL download. The commands are listed within each example under the different “Figure” listings, showing the SQL required to generate the query, along with the results returned.

The examples show typical ways that the data can be manipulated and queried from the perspective of analysing historic results. In the next section, *Querying the Daily Tables*, we cover examples from the perspective of queries on upcoming races and recent results.

Example 1: Looking up a runner’s racing history

This example shows how to retrieve details of a particular runner’s racing history in the UK and Ireland by specifying the runner’s name. This type of query is useful for ad-hoc research or more typically if looking up certain historic attributes of each competing horse from a daily racecard. Combined with the **daily_runners** table, such queries can also be automated to retrieve the full history of all runners in an upcoming race, and rate them according to the attributes found in the database against the user’s own criteria.

This example and subsequent ones show how almost any useful query on historic data in the SmartForm database combines information from both the **historic_races** and **historic_runners** tables. Since joint information from two tables is commonly used in all queries, we will have to specify how we want the tables to be joined.

The typical JOIN is always on **race_id** between **historic_runners** and **historic_tables**. Taking the example of the most basic information from a runner’s history, we will want to see the **meeting_date** (found in the **historic_races** table), and the **finish_position** (found in the **historic_runners** table) so we need the **race_id** for both items of data from the different tables to be the same – otherwise the query will not make sense. (Note that the WHERE condition can also be used to achieve this, and is shown later.)

A basic query to find all the runs of Sir Percy, the 2006 Derby winner, and display the basic variables above, reads as per Figure 4.1.1, with the results of the query also shown:

```
mysql> SELECT meeting_date, finish_position from historic_races join historic_runners using
(race_id) where name = "Sir Percy";

+--------------+-----------------+
| meeting_date | finish_position |
+--------------+-----------------+
| 2005-05-28   |               1 |
| 2005-06-23   |               1 |
| 2005-07-27   |               1 |
| 2005-10-15   |               1 |
| 2006-06-03   |               1 |
| 2006-05-06   |               2 |
| 2006-10-14   |               7 |
| 2007-06-01   |               6 |
| 2007-06-20   |               6 |
+--------------+-----------------+
9 rows in set (0.00 sec)
```

Figure 4.1.1: Basic query for Sir Percy's results

The results table from Figure 4.1.1 provides scant information, is not strictly in date order, and the column names are taking up too much width for the information shown.

We can address the formatting issues first, by choosing a display name for the column with the command AS 'display name', and ranking the query strictly by date (by using ORDER BY, then the field to order the returned rows by, in this case, date). We'll also change the default order so that we see most recent runs first, using the DESC to indicating descending order, as follows:

```
mysql> SELECT meeting_date AS 'Date', finish_position AS 'Pos' from historic_races join
historic_runners using (race_id) where name = "Sir Percy" ORDER BY date DESC;

+------------+------+
| Date       | Pos  |
+------------+------+
| 2007-06-20 |    6 |
| 2007-06-01 |    6 |
| 2006-10-14 |    7 |
| 2006-06-03 |    1 |
| 2006-05-06 |    2 |
| 2005-10-15 |    1 |
| 2005-07-27 |    1 |
| 2005-06-23 |    1 |
| 2005-05-28 |    1 |
+------------+------+
9 rows in set (0.00 sec)
```

Figure 4.1.2: Basic query from Figure 4.1.1 reformatted

That's a lot better in terms of format, but still not very informative with regard to each of Sir Percy's runs. In terms of adding further information, this is purely down to the user's requirements at this point, since the query for adding data is of the same format as the basic query. Any or all of the fields in **historic_runners** and **historic_races**, as listed in Section 2, can be listed within the query to provide more data, as shown in Figure 4.1.3.

Thus we add more values in Figure 4.1.3, showing the extended MySQL query and results from the MySQL client. The fields were chosen from the tables in Section 2 as follows:

- Date (**meeting_date** field, in **historic_races**)
- Course (**course** field, in **historic_races**)
- Finishing position (**finish_position** field, in **historic runners**)
- Number of horses ran (**num_finishers** field in **historic_races**)
- Distance of the race in yards (**distance_yards** in **historic_races**)
- Going description (**going** in **historic_races**)
- Class of race (**class** in **historic_races**)
- Distance beaten by winner (**distance_behind** winner in **historic_runners**)
- Starting price (the **starting_price** in fractions format in **historic_runners**)

Note that in the query in Figure 4.1.3 we also change the query slightly to show use of the WHERE condition for joining the 2 tables as an alternative to JOIN, which still joins tables on field common to both (namely **race_id**):

Once you have a feel for the basic way that the tables are organised – anything specific to a runner is in historic_runners, and anything specific to a race is in historic_races, knowing where to look for the fields is quite intuitive. After that, it's a question of becoming familiar with the field names. The autocomplete function in the MySQL client can serve as a useful reminder without referring back to the manual.

```
mysql> SELECT meeting_date AS 'Date', course, finish_position AS 'Pos', num_finishers AS
'Ran ', distance_yards AS 'Dist', going, class, distance_behind_winner AS 'lost_by',
starting_price as 'SP' from historic_races, historic_runners where name = "Sir Percy" and
(historic_races.race_id = historic_runners.race_id) ORDER BY Date DESC;

+------------+-------------+------+------+------+--------------+-------+---------+------+
| Date       | course      | Pos  | Ran  | Dist | going        | class | lost_by | SP   |
+------------+-------------+------+------+------+--------------+-------+---------+------+
| 2007-06-20 | Ascot       |    6 |    6 | 2200 | Good         |     1 |    8.25 | 11/1 |
| 2007-06-01 | Epsom_Downs |    6 |    7 | 2640 | Good to Soft |     1 |    5.88 | 7/2  |
| 2006-10-14 | Newmarket   |    7 |    8 | 2200 | Good to Soft |     1 |    7.69 | 11/4 |
| 2006-06-03 | Epsom_Downs |    1 |   17 | 2650 | Good to Firm |     1 |    NULL | 6/1  |
| 2006-05-06 | Newmarket   |    2 |   14 | 1760 | Good to Firm |     1 |    2.50 | 4/1  |
| 2005-10-15 | Newmarket   |    1 |    8 | 1540 | Good to Soft |     1 |    NULL | 9/2  |
| 2005-07-27 | Goodwood    |    1 |    7 | 1540 | Soft         |     1 |    NULL | 4/1  |
| 2005-06-23 | Salisbury   |    1 |    8 | 1320 | Good to Firm |     2 |    NULL | 7/4  |
| 2005-05-28 | Goodwood    |    1 |   11 | 1320 | Good         |     5 |    NULL | 8/1  |
+------------+-------------+------+------+------+--------------+-------+---------+------+
9 rows in set (0.00 sec)
```

Figure 4.1.3: Fuller Race History for Sir Percy

Looking purely at race history and finishing order ordered in this way, we can start to better understand Sir Percy's record. At first glance, we can see that his performances seem to have deteriorated after the Epsom Derby on 03/06/2006.

Up to and including the Epsom Derby, Sir Percy had won or come second in every race, after the Epsom Derby, he came last or second last in every race. There is no obvious explanation for deterioration of comparative form in later races, since the other race characteristics in the poorer races do not seem to be significantly different from the race characteristics in the better races – eg. different types of going and distance had been successfully encountered in the few runs up to and including the Derby, as well as afterwards. However, for a horse with a longer running history, it would be worth filtering runs purely on going or course by adding another condition to the WHERE part of this query – we will see cases of this syntax in subsequent examples.

Of course, there are many other critical variables that could be retrieved in order to analyse running history further, not least some absolute measures of performance, such as official ratings. Generally, any other variable can be added to or subtracted from an existing query in the MySQL client very easily. You can use the Up and Down keys to arrive at any previous command typed into the mysql client during the current session. That can then be edited you want to adapt, and then using the autocomplete function hit the TAB key) to help complete names of fields.

Example 2: Analysing draw bias/significance and profitability

Here we look at the process of using the database for research into the significance of specific variables, in this case, the significance of the draw in flat racing.

Typically, draw bias is not used alone as a betting system, any more than any other single variable, but rather as one, possibly very influential, factor within a betting system or ratings model. In deciding what variables to use within such systems and models, the process of quantifying the significance of the draw is generally applicable to other variables in the database.

For this particular example, we concentrate on looking at one of the more well known draw biases at Chester, where the tight turns on the track would seem to offer an obvious advantage horses drawn closer to the inside rail. However, it is still to be determined exactly what that advantage is, and how profitable is following the draw alone.

The effect of the draw is of course different for different tracks, and for those tracks, can vary greatly in its significance according to different conditions - with race distance, number of runners and going the most widely acknowledged contributing factors. Using the SmartForm database provides a way to assess the effect of the draw for particular tracks under different conditions on an ongoing basis.

So, let’s look at the process and start to run some specific queries:

To begin to analyse the significance of a horse being drawn in any particular part of the track, we first have to look at the number of races at the particular course, to see if there are a significant number of races to query. Given that we are interested in the draw bias at Chester, identifying the number of races run at Chester is the first query.

```
mysql> select count(*) from historic_races where course="Chester";
```

```
+----------+
| count(*) |
+----------+
|      445 |
+----------+
1 row in set (0.00 sec)
```

Figure 4.2.1: Number of races found for Chester between 01/01/03 and 01/09/08

The COUNT function applied to any variable returns the number of rows (or times occurred) found for the query, as opposed to individual results. It's very useful in many of the operations for which the database is typically used, since assessing systems or the significance of variables often involves identifying and measuring (ie. counting) the number of times an event has occurred - relative to the possible number of outcomes. We'll also see this in looking at profitability of certain events.

So far, all we have done is count the number of races run. Next, given that distance is a key variable in assessing the effect of the draw, let's break down these races by distance, and start to look at the distribution of stall numbers in terms of which stall the winning horse came from. To do this, we show all unique distances run at Chester, using the COUNT function to count each unique distance, in conjunction with GROUP BY to itemise each unique distance. To show how significant the draw is for each distance found, we use the average function (AVG) applied to both the stall number and the number of finishers in a race, to show the average winning stall number and average field size for each distance.

```
mysql> select distance_yards, count(distance_yards), AVG(stall_number), AVG(num_finishers)
from historic_races join historic_runners using (race_id) where course="Chester" and
finish_position=1 GROUP BY distance_yards;

+----------------+-----------------------+-------------------+--------------------+
| distance_yards | count(distance_yards) | AVG(stall_number) | AVG(num_finishers) |
+----------------+-----------------------+-------------------+--------------------+
|           1100 |                    31 |            3.9677 |             8.8710 |
|           1116 |                    65 |            3.4615 |             9.9077 |
|           1320 |                    16 |            3.0000 |             7.4375 |
|           1338 |                    29 |            4.0000 |             9.0345 |
|           1540 |                    22 |            4.0909 |             9.7727 |
|           1542 |                    47 |            4.9362 |             9.1702 |
|           1662 |                    34 |            6.2647 |            10.8529 |
|           1760 |                    17 |            6.4706 |            11.2941 |
|           2200 |                    32 |            4.0000 |             8.6563 |
|           2275 |                    54 |            4.9815 |             8.9815 |
|           2420 |                     2 |            4.0000 |            10.0000 |
|           2499 |                     4 |            3.0000 |             9.0000 |
|           2640 |                    19 |            4.7895 |             8.9474 |
|           2706 |                    31 |            5.1290 |             9.0323 |
|           2860 |                     5 |            4.6000 |             9.6000 |
|           2949 |                     9 |            3.4444 |             8.2222 |
|           3495 |                    12 |            5.7500 |            10.2500 |
|           3520 |                     5 |            5.0000 |             7.2000 |
|           4107 |                     6 |            5.0000 |            13.0000 |
|           4180 |                     2 |           13.5000 |            16.0000 |
+----------------+-----------------------+-------------------+--------------------+
20 rows in set (0.00 sec)
```

Figure 4.2.2: Winning average stall number to average finishers, grouped by distance

At any one particular distance the numbers are not hugely informative, but as a group there are some notable points. For example, all the winners under 1540 yards, at 4 specific distances, accounting basically for 5 and 6 furlong races, come out on average in the lower half of the draw. If we run the same query to look purely at horses in this distance range, aggregated together, this statistic is more apparent.

```
mysql> select count(distance_yards), AVG(stall_number), AVG(num_finishers) from
historic_races join historic_runners using (race_id) where course="Chester" and
finish_position=1 and distance_yards < 1500;

+-----------------------+-------------------+--------------------+
| count(distance_yards) | AVG(stall_number) | AVG(num_finishers) |
+-----------------------+-------------------+--------------------+
|                   141 |            3.6312 |             9.2199 |
+-----------------------+-------------------+--------------------+
1 row in set (0.26 sec)
```

Figure 4.2.3: Average winning stall for different field sizes in sprint races

Of course, averages can be misleading. There cannot be a draw of 3.6 any more than 9.2 runners can contend the race. These averages also include results of 3 runner races, where we would not expect the draw to have any significance, with 16 runner races, where we would. Therefore, there are further queries to look at before we can arrive at any sensible results.

Typically, it is safe to assume that stall bias is more apparent when there are minimum number of runners in the race. This is for several reasons, mainly that no horse is forced to race significantly far away from the other horses in a small field, so draw bias is more difficult to detect. This assumption can of course be tested itself. Let's therefore split the last query on draw bias up according to the number of finishers in the race, as shown in Figure 4.2.4.

```
mysql> select num_finishers, count(num_finishers) AS 'Races', AVG(stall_number) from
historic_races join historic_runners using (race_id) where course="Chester" and
finish_position=1 and distance_yards < 1500 GROUP BY num_finishers;

+---------------+----------------------+-------------------+
| num_finishers |      Races           | AVG(stall_number) |
+---------------+----------------------+-------------------+
|             3 |                    2 |            4.5000 |
|             4 |                    6 |            2.3333 |
|             5 |                    7 |            3.0000 |
|             6 |                   13 |            3.1538 |
|             7 |                   13 |            3.3077 |
|             8 |                   18 |            3.6111 |
|             9 |                   21 |            3.8571 |
|            10 |                   20 |            3.5500 |
|            11 |                    8 |            4.6250 |
|            12 |                    6 |            4.8333 |
|            13 |                   12 |            4.4167 |
|            14 |                    9 |            2.1111 |
|            15 |                    5 |            4.8000 |
|            16 |                    1 |            5.0000 |
+---------------+----------------------+-------------------+
14 rows in set (0.24 sec)
```

Figure 4.2.4: Average winning stall for different field sizes in sprint races

Given that the draw results from higher numbers of finishers should be more indicative of any significant bias it typically therefore makes sense to apply a minimum number of runners to detecting stall bias. We can see in the results where any significance might be and where it might make sense to do so, with a possible rule of thumb being that the lower the average draw number and the higher the field size, the greater the significance.

Here is a final adapted query which shows the average ratio of average runners to the average winning stall number:

```
mysql> select COUNT(num_finishers), AVG(stall_number), AVG(num_finishers),
AVG(stall_number)/num_finishers AS 'ratio'  from historic_races join historic_runners using
(race_id) where course="Chester" and finish_position=1 and distance_yards < 1500 and
num_finishers > 9;
+----------------------+-------------------+--------------------+------------+
| COUNT(num_finishers) | AVG(stall_number) | AVG(num_finishers) | ratio      |
+----------------------+-------------------+--------------------+------------+
|                   61 |            3.9016 |            12.0164 | 0.26010929 |
+----------------------+-------------------+--------------------+------------+
1 row in set (0.23 sec)
```

Figure 4.2.5: Average winning stall for fields of 10 and more in Chester sprints

A ratio of 0.26 over 61 races looks fairly significant. In other words, the winner of every race came on average from the top quarter of the draw. However, none of the queries so far has shown the results of the stall bias directly or whether or not betting on that apparent bias might be profitable.

Therefore, in Figure 4.2.6, let’s begin this process by breaking down these 61 races according to their winning stall number and add in the average starting price for each winner from that stall.

The “count(stall_number)” column shows the number of times this stall number won for races over 10 runners at distances of less than 7 furlongs. This itself looks highly correlated with the draw, suggesting that it is a significant factor.

```
mysql> select stall_number, count(stall_number), AVG(starting_price_decimal)  from
historic_races join historic_runners using (race_id) where course="Chester" and
finish_position=1 and distance_yards < 1500 and num_finishers > 9 GROUP BY stall_number;

+--------------+---------------------+-----------------------------+
| stall_number | count(stall_number) | AVG(starting_price_decimal) |
+--------------+---------------------+-----------------------------+
|            1 |                  15 |                    7.449286 |
|            2 |                  10 |                    6.975000 |
|            3 |                   8 |                    6.375000 |
|            4 |                   7 |                    8.313333 |
|            5 |                   5 |                    7.140000 |
|            6 |                   5 |                   12.100000 |
|            7 |                   3 |                    9.000000 |
|            8 |                   3 |                   12.000000 |
|            9 |                   1 |                    8.000000 |
|           10 |                   3 |                   13.666667 |
|           12 |                   1 |                   17.000000 |
+--------------+---------------------+-----------------------------+
11 rows in set (0.26 sec)
```

Figure 4.2.6: Winning stall counts in sprint races over 10 runners, with average SPs

Let’s tidy this query up a little (in terms of aliasing the columns to make the names shorter and more meaningful) and, moreover, look at the overall (historic) profitability of backing each draw blind in sprints at Chester with larger fields.

Since MySQL queries, as we have already shown, can incorporate calculations on the values returned, we can do this within the context of the new query, as in Figure 4.2.7, by multiplying the winners from each stall by the average price of winners from that stall, then subtracting the total number of bets, assuming we would have had one bet on each stall in every race, for all 61 races (given that stall 11 and 12 was not present in every race, this is not an accurate representation of backing those stalls blind, of course).

```
mysql> select stall_number, count(stall_number) AS 'winners', AVG(starting_price_decimal)
AS 'average_price', count(stall_number)*AVG(starting_price_decimal)-61 as 'Profit'  from
historic_races join historic_runners using (race_id) where course="Chester" and
finish_position=1 and distance_yards < 1500 and num_finishers > 9 GROUP BY stall_number;

+--------------+---------+---------------+------------+
| stall_number | winners | average_price | Profit     |
+--------------+---------+---------------+------------+
|            1 |      15 |      7.449286 |  50.739286 |
|            2 |      10 |      6.975000 |   8.750000 |
|            3 |       8 |      6.375000 | -10.000000 |
|            4 |       7 |      8.313333 |  -2.806667 |
|            5 |       5 |      7.140000 | -25.300000 |
|            6 |       5 |     12.100000 |  -0.500000 |
|            7 |       3 |      9.000000 | -34.000000 |
|            8 |       3 |     12.000000 | -25.000000 |
|            9 |       1 |      8.000000 | -53.000000 |
|           10 |       3 |     13.666667 | -20.000000 |
|           12 |       1 |     17.000000 | -44.000000 |
+--------------+---------+---------------+------------+
11 rows in set (0.34 sec)
```

Figure 4.2.7: Profitability of stall bias in Chester sprint races

In conclusion, backing stalls 1 or 2 blind over all races produces a significant profit. A point to note for testing profitability of systems or variables in general is the use of the starting_price_decimal field, as opposed to starting price, for ease of calculation.

Finally, let’s start to look at whether profitability varies over time by adding in a date condition, namely showing the query in Figure 4.2.8 but with the addition of splitting results where the meeting date is greater than 1st January 2006 `meeting_date > "2006-01-01"`. This is roughly half way through the historic database. The number of races produced are also approximately half, at 27 races. (Note that the profitability is therefore calculated off the basis of betting one unit point in all 27, rather than 61 races).

```
mysql> select stall_number, count(stall_number) AS 'winners', AVG(starting_price_decimal)
AS 'average_price', count(stall_number)*AVG(starting_price_decimal)-27 AS 'Profit'  from
historic_races join historic_runners using (race_id) where course="Chester" and
finish_position=1 and distance_yards < 1500 and meeting_date > "2006-01-01" and
num_finishers > 9 GROUP BY stall_number;
+--------------+---------+---------------+------------+
| stall_number | winners | average_price | Profit     |
+--------------+---------+---------------+------------+
|            1 |       6 |      7.105000 |  15.630000 |
|            2 |       6 |      7.500000 |  18.000000 |
|            3 |       2 |      8.000000 | -11.000000 |
|            4 |       3 |      4.793333 | -12.620000 |
|            5 |       2 |      9.000000 |  -9.000000 |
|            6 |       3 |     11.500000 |   7.500000 |
|            7 |       1 |     13.000000 | -14.000000 |
|            8 |       1 |      6.000000 | -21.000000 |
|            9 |       1 |      8.000000 | -19.000000 |
|           10 |       2 |     12.000000 |  -3.000000 |
+--------------+---------+---------------+------------+
10 rows in set (0.27 sec)
```

Figure 4.2.8: Profitability of different stalls in Chester sprint races with 10 or more runners after 01/01/2006

As we can see, the system is still profitable in recent years. since 2006, with stalls 1 and 2 continuing to return a profit through backing them blind, without any consideration as to the ability, preferences or form of the horse, jockey or trainer.

In the next section, we will continue this example to look at how to apply rules representing a system such as this to the daily racecards on an ongoing basis.

Querying Daily Tables

This section runs through some examples which use the Daily Tables, namely **daily_races** and **daily_runners**. These tables are updated automatically with the forthcoming day's racecards (available from the night before racing), as well as the previous day's results, to ensure that the historic tables are up to date. Users can typically query these tables on ad-hoc basis and apply systems or models to daily data to identify potential bets, or to rate all races according to an existing model.

In the examples below, we produce fewer ad hoc queries on the data to get to the required results since we assume that the examples already shown have been read and therefore that the basic approach and syntax to querying the database is understood.

As with the queries on historic_runners and historic_runners, queries on daily races generally involve a join on two tables, in this case daily_races and daily_runners, which are similarly split by race attributes and runner attributes.

Example 3: Retrieve summary race details for daily cards by user criteria

Example 3 and Example 4 show how we can use the daily tables to interrogate the day's racing on an ad-hoc basis.

Example 3 is a query to list all races in the UK and Ireland for the upcoming day according to certain criteria determined by the user – such as whether or not the race falls in a certain race type, where races are over a certain value, and so on. Any of the criteria in daily_races can be used as criteria.

With the database come a set of previous examples of daily_races and daily_runners files (the up to date versions of these are supplied as part of the daily updates service), and it is those we use for this example. Imagining that the current date is 7th June 2008, which just happens to be Derby Day, we can produce the following races that are due off today by simply selecting a few attributes, as shown in the query, that are pertinent to the racing.

```
mysql> select course, scheduled_time, race_type, distance_yards, added_money from
daily_races where meeting_date="2008-06-07";
+-------------+---------------------+-----------+----------------+-------------+
| course      | scheduled_time      | race_type | distance_yards | added_money |
+-------------+---------------------+-----------+----------------+-------------+
| Epsom_Downs | 2008-06-07 13:40:00 | Flat      |           2218 |    50000.00 |
| Epsom_Downs | 2008-06-07 14:10:00 | Flat      |           1320 |    25000.00 |
| Epsom_Downs | 2008-06-07 14:40:00 | Flat      |           1874 |    50000.00 |
| Epsom_Downs | 2008-06-07 15:15:00 | Flat      |           1100 |    50000.00 |
| Epsom_Downs | 2008-06-07 16:00:00 | Flat      |           2650 |  1000000.00 |
| Epsom_Downs | 2008-06-07 16:45:00 | Flat      |           2650 |    25000.00 |
| Epsom_Downs | 2008-06-07 17:20:00 | Flat      |           1320 |    25000.00 |
| Musselburgh | 2008-06-07 14:20:00 | Flat      |           1100 |    20000.00 |
| Musselburgh | 2008-06-07 14:50:00 | Flat      |           3080 |    15000.00 |
| Musselburgh | 2008-06-07 15:20:00 | Flat      |           2640 |    25000.00 |
| Musselburgh | 2008-06-07 15:50:00 | Flat      |           1760 |     7500.00 |
| Musselburgh | 2008-06-07 16:35:00 | Flat      |           1570 |     7500.00 |
| Musselburgh | 2008-06-07 17:05:00 | Flat      |           1100 |     8000.00 |
| Musselburgh | 2008-06-07 17:35:00 | Flat      |           3520 |     6000.00 |
| Lingfield   | 2008-06-07 17:40:00 | Flat      |           2640 |     2600.00 |
| Lingfield   | 2008-06-07 18:10:00 | Flat      |           2200 |     3000.00 |
| Lingfield   | 2008-06-07 18:40:00 | Flat      |           1760 |     3600.00 |
| Lingfield   | 2008-06-07 19:10:00 | Flat      |           1320 |     5100.00 |
| Lingfield   | 2008-06-07 19:40:00 | Flat      |           1100 |     3600.00 |
| Lingfield   | 2008-06-07 20:10:00 | Flat      |           1540 |     3600.00 |
| Curragh     | 2008-06-07 17:30:00 | Flat      |           1320 |    20001.00 |
| Curragh     | 2008-06-07 18:00:00 | Flat      |           1320 |    16000.00 |
| Curragh     | 2008-06-07 18:30:00 | Flat      |           1320 |    11000.00 |
| Curragh     | 2008-06-07 19:00:00 | Flat      |           2200 |    55001.00 |
| Curragh     | 2008-06-07 19:30:00 | Flat      |           2200 |    13500.00 |
| Curragh     | 2008-06-07 20:00:00 | Flat      |           1760 |    27000.00 |
| Curragh     | 2008-06-07 20:30:00 | Flat      |           1760 |    15000.00 |
| Newcastle   | 2008-06-07 18:50:00 | Flat      |           1320 |     4500.00 |
| Newcastle   | 2008-06-07 19:20:00 | Flat      |           1540 |     5000.00 |
| Newcastle   | 2008-06-07 19:50:00 | Flat      |           2232 |     4500.00 |
| Newcastle   | 2008-06-07 20:20:00 | Flat      |           2232 |     4500.00 |
| Newcastle   | 2008-06-07 20:50:00 | Flat      |           3539 |     4000.00 |
| Newcastle   | 2008-06-07 21:20:00 | Flat      |           1100 |     4500.00 |
| Hexham      | 2008-06-07 14:15:00 | Hurdle    |           4510 |     3600.00 |
| Hexham      | 2008-06-07 14:45:00 | Chase     |           4510 |     4500.00 |
| Hexham      | 2008-06-07 15:25:00 | Chase     |           5500 |     4500.00 |
| Hexham      | 2008-06-07 16:10:00 | Hurdle    |           4510 |     4000.00 |
| Hexham      | 2008-06-07 16:40:00 | Chase     |           4510 |     1800.00 |
| Hexham      | 2008-06-07 17:10:00 | Hurdle    |           3630 |     4000.00 |
| Worcester   | 2008-06-07 13:50:00 | Chase     |           5060 |     5000.00 |
| Worcester   | 2008-06-07 14:25:00 | Hurdle    |           4400 |     3600.00 |
| Worcester   | 2008-06-07 14:55:00 | Chase     |           3520 |     5000.00 |
| Worcester   | 2008-06-07 15:35:00 | Hurdle    |           4400 |     7000.00 |
| Worcester   | 2008-06-07 16:20:00 | Chase     |           4510 |     9000.00 |
| Worcester   | 2008-06-07 16:50:00 | Hurdle    |           5280 |     4000.00 |
| Worcester   | 2008-06-07 17:25:00 | Hurdle    |           4400 |     3600.00 |
| Doncaster   | 2008-06-07 14:05:00 | Flat      |           1540 |    15000.00 |
| Doncaster   | 2008-06-07 14:35:00 | Flat      |           1760 |    20000.00 |
| Doncaster   | 2008-06-07 15:05:00 | Flat      |           2260 |    25000.00 |
| Doncaster   | 2008-06-07 15:45:00 | Flat      |           1320 |     6000.00 |
| Doncaster   | 2008-06-07 16:30:00 | Flat      |           2640 |     7500.00 |
| Doncaster   | 2008-06-07 17:00:00 | Flat      |           1100 |     7000.00 |
| Doncaster   | 2008-06-07 17:30:00 | Flat      |           1100 |     7000.00 |
+-------------+---------------------+-----------+----------------+-------------+
53 rows in set (0.03 sec)
```

Figure 4.3.1: Displaying all daily races by user specified criteria

The power of this type of query for identifying races of interest immediately becomes apparent when we start to apply filters. In this case, we may only be interested in betting races over a certain prize money, distance or race type. Now we can immediately identify such races, simply by adding a WHERE clause and a condition, such as WHERE handicap=1, in the case of wanting to modify the query in Figure 4.3.1 such that only handicap races are returned for consideration.

Example 4: Create detailed racecard for daily racing

This example shows how to display full racecards for any particular race on the daily cards

Of course, with the SmartForm database the definition of full racecards will be determined by the user, since there are over 50 fields in daily_runners and more than that to account for in daily_races.

Here we have gone for basic information that fits within one page width (many more rows can be successfully displayed on one line from the MySQL monitor client. We also specify an obvious filter for the date in question in terms of which race to use (ie. we choose the only race paying over £900,000 in prizemoney) for building the card, all of which can easily be reconfigured.

```
mysql> select scheduled_time, name, form_figures AS 'form', jockey_name, trainer_name from
daily_runners join daily_races using (race_id) where meeting_date="2008-06-07" and
added_money > 900000 and course="Epsom_Downs";
+---------------------+-------------------+---------+---------------+--------------------+
| scheduled_time      | name              | form    | jockey_name   | trainer_name       |
+---------------------+-------------------+---------+---------------+--------------------+
| 2008-06-07 16:00:00 | Alan Devonshire   | 61223-4 | P Mulrennan   | M H Tompkins       |
| 2008-06-07 16:00:00 | Alessandro Volta  | 831-41  | J A Heffernan | A P O'Brien        |
| 2008-06-07 16:00:00 | Bashkirov         | 34-53   | D R McCabe    | A P O'Brien        |
| 2008-06-07 16:00:00 | Bouguereau        | 743-127 | A Munro       | P W Chapple-Hyam   |
| 2008-06-07 16:00:00 | Casual Conquest   | 1-1     | P J Smullen   | D K Weld           |
| 2008-06-07 16:00:00 | Curtain Call      | 62215-1 | J P Spencer   | L M Cumani         |
| 2008-06-07 16:00:00 | Doctor Fremantle  | 221-21  | K McEvoy      | Sir Michael Stoute |
| 2008-06-07 16:00:00 | Frozen Fire       | 18-2    | M J Kinane    | A P O'Brien        |
| 2008-06-07 16:00:00 | Kandahar Run      | 211-21  | T E Durcan    | H R A Cecil        |
| 2008-06-07 16:00:00 | King of Rome      | 710-52  | J Murtagh     | A P O'Brien        |
| 2008-06-07 16:00:00 | Maidstone Mixture | 00-5303 | M O'Connell   | Paul Murphy        |
| 2008-06-07 16:00:00 | New Approach      | 1111-22 | K J Manning   | J S Bolger         |
| 2008-06-07 16:00:00 | Rio de La Plata   | 11214-2 | L Dettori     | Saeed bin Suroor   |
| 2008-06-07 16:00:00 | River Proud       | 2817-43 | T Quinn       | P F I Cole         |
| 2008-06-07 16:00:00 | Tajaaweed         | 10-1    | R Hills       | Sir Michael Stoute |
| 2008-06-07 16:00:00 | Tartan Bearer     | 2-11    | R L Moore     | Sir Michael Stoute |
| 2008-06-07 16:00:00 | Washington Irving | 4-22    | C O'Donoghue  | A P O'Brien        |
+---------------------+-------------------+---------+---------------+--------------------+
17 rows in set (0.48 sec)17 rows in set (0.41 sec)
```

Figure 4.4.1: Racecard for Derby Day configured by user

As with the queries on historic_runners and historic_runners, queries on daily races generally involve a join on two tables, in this case daily_races and daily_runners, which are analogous to the type of data contained in the historic database, again split by race attributes and runner attributes.

Example 5: Applying a system or model to daily racing

Earlier we saw the example from the subsection Querying Historic Tables which investigated and found significant draw bias in certain race types at Chester. This was interesting research, but how could such a system be applied to make money on a daily basis, in an automated way?

Example 5 shows one way of doing that. Example 2 provides us with the clue we need as to rules.

The query is as follows: taken as a group, all horses drawn in stalls 1 and 2 show long term profitability when racing over Chester sprint races. Therefore we need to specify these criteria for the purposes of generating latest selections.

Thus, if we imagine that the current date is 12th July 2008, and that this query is run before racing, then the query shown in Figure 4.5.1 is appropriate to find qualifying horses, as foolows:

```
mysql> select scheduled_time, name, form_figures 'AS', jockey_name, trainer_name from
daily_runners join daily_races using (race_id) where meeting_date="2008-07-12" and
course="Chester" and (stall_number=1 OR stall_number=2) and distance_yards < 1400;
+---------------------+------------------+---------+-------------+--------------+
| scheduled_time      | name             | AS      | jockey_name | trainer_name |
+---------------------+------------------+---------+-------------+--------------+
| 2008-07-12 15:35:00 | Green Manalishi  | 0465-63 | E Ahern     | K A Ryan     |
| 2008-07-12 15:35:00 | Angus Newz       | 230611  | M Fenton    | M Quinn      |
| 2008-07-12 16:10:00 | Red Baron Dancer | 22      | P Cosgrave  | J R Boyle    |
| 2008-07-12 16:10:00 | Dark Velvet      | 7       | C Catlin    | E J Alston   |
+---------------------+------------------+---------+-------------+--------------+
4 rows in set (0.35 sec)
```

Figure 4.5.1: Selecting qualifiers for a daily system

This should be fairly intuitive. The query specifies the conditions for the system against the daily cards in an attempt to look for qualifying candidates within the system, and profitable to back horses. Therefore, using the date, the course name, whether the stall either 1 and stall 2, we can find qualifying horses.

We see results for 4 horses in the returned results. This stands to reason, since there are two sprint races on the day, so it follows that stalls 1 and 2 will be occupied, therefore throwing up 4 bets. It turns out that Green Manalishi won its race. Indeed, the system was profitable over the whole of 2008, and using this query from SmartForm identifies the winner each time.

Appendix 1: Additional MySQL references

This short list is aimed at the beginner to MySQL. Many more references are available and can be found online, but the below already cover more than enough material, with the recommended starting text being *Learning MySQL.*

The material outlined in *Learning MySQL* is also sufficient to cover any of the described uses of the SmartForm database and much more. Thereafter, the references become slightly more esoteric in terms of optimising database performance and covering programmatic access to the database.

See also online resources for MySQL at www.mysql.com

Learning MySQL
Seyed M.M. Tahaghoghi & Hugh E. Williams
O'Reilly 2006

MySQL Third Edition
Paul DuBois
Sams, 2005

MySQL Cookbook
Paul DuBois
O'Reilly, 2007